ISO TC184/SC4 N536

1997-03-01

ISO TC 184/SC4 STANDING DOCUMENT

Technical Committee 184 for Industrial Automation Systems and Integration

Subcommittee 4 for Industrial Data

Guidelines for the development of abstract test suites

SC4 Secretariat
National Institute of Standards and Technology
Building 220/Room A127
Gaithersburg, Maryland 20899
USA

Contents

Annexes

Figures

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

This standing document was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC4, *Industrial data*.

ISO/TC 184/SC4 standards are prepared according to guidelines put forth in the following standing documents:

— Guidelines for application interpreted construct development;

— Guidelines for application interpreted model development;

— Guidelines for the development and approval of STEP application protocols;

— Guidelines for the development of abstract test suites;

— Guidelines for the development of mapping tables;

— ISO/TC 184/SC4 organization handbook;

— Supplementary directives for the drafting and presentation of ISO 10303.

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective it to provide a neutral mechanism capable of describing product data throughout the life cycle of a product independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.

This International Standard is organized as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series: description methods, integrated resources, application interpreted constructs, application protocols, abstract test suites, implementation methods, and conformance testing. The series are described in ISO 10303-1.

The purpose of this document is to provide methods and procedures for the development of abstract test suites for conformance testing products against ISO 10303 application. The intended audience for this document is application protocol developers and abstract test suite developers. These abstract test suites will be developed by developers of the respective ISO 10303 application protocols.

The contents of this document reflect the experience of development and review of ISO 10303-301, ISO 10303-302, ISO 10303-303, ISO 10303-304, ISO 10303-307, ISO 10303-312, ISO 10303-313, ISO 10303-314, ISO 10303-325, and ISO 10303-327 as well as the results of numerous discussions among members of the ISO 10303 community.

A list of tools supporting the development of abstract test suites can be found on the STEP On-Line Information Service (SOLIS).

Guidelines for the development of abstract test suites

1 Scope

This standing document sets out methods and procedures for the development and documentation of abstract test suites for STEP application protocols. Each application protocol is associated with a corresponding abstract test suite.

The following are within the scope of this document:

— the organization and documentation of an abstract test suite suitable for use in the testing of implementations based on ISO 10303-21 and ISO 10303-22;

— the development and formatting of test purposes;

— the development and formatting of verdict criteria;

— the formatting of abstract test cases;

— suggested approaches for the structuring and creation of abstract test cases;

— suggested approaches for the validation of an abstract test suite and its component abstract test cases.

The following are outside the scope of this document:

— methods or procedures for the development of conformance requirements of application protocols;

— methods or procedures for the partitioning of application protocols into its conformance classes;

— methods or procedures for the evaluation of interoperability between implementations based on different application protocols.

2 Normative references

The following standing documents and standards contain provisions which, through reference in this text, constitute provisions of this standing document. At the time of publication, the editions indicated were valid. All standing documents and standards are subject to revision, and parties to agreements based on this standing document are encouraged to investigate the possibility of applying the most recent editions of the standing documents and standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 10303-1:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and fundamental principles.*

ISO 10303-11:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 11: Descriptive methods: The EXPRESS language reference manual.*

ISO 10303-21:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure.*

ISO 10303-22:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 22: Implementation methods: Standard data access interface.*

ISO 10303-31:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 31: Conformance testing methodology and framework: General concepts.*

ISO 10303-32[2)], *Industrial automation systems and integration - Product data representation and exchange - Part 32: Conformance testing methodology and framework: Requirements on testing laboratories and clients.*

ISO 10303-34[2)], *Industrial automation systems and integration - Product data representation and exchange - Part 34: Conformance testing methodology and framework: Abstract test methods.*

ISO 10303-35[2)], *Industrial automation systems and integration - Product data representation and exchange - Part 35: Conformance testing methodology and framework: Abstract test methods for the standard data access interface.*

ISO/IEC8824-1:1994, *Information technology - Open systems interconnection - Abstract syntax notation one (ASN.1) - Part 1: Specification of basic notation.*

3 Definitions and abbreviations

3.1 Terms defined in ISO 10303-1

This standing document makes use of the following terms defined in ISO 10303-1:

3.1.1 abstract test suite: a part of this International Standard that contains the set of abstract test cases necessary for conformance testing of an implementation of an application protocol.

3.1.2 application interpreted model (AIM): an information model that uses the integrated resources necessary to satisfy the information requirements and constraints of an application reference model, within an application protocol.

3.1.3 **application protocol (AP):** a part of this International Standard that specifies an application interpreted model satisfying the scope and information requirements for a specific application.

3.1.4 **application reference model (ARM):** an information model that describes the information requirements and constraints of a specific application context.

3.1.5 **application object:** an atomic element of an application reference model that defines a unique concept of the application and contains attributes specifying the data elements of the object.

3.1.6 **conformance class:** a subset of an application protocol for which conformance may be claimed.

3.1.7 **exchange structure:** a computer-interpretable format used for storing, accessing, transferring, and archiving data.

3.1.8 **implementation method:** a part of this International Standard that specifies a technique used by computer systems to exchange product data that is described using the EXPRESS data specification language [ISO 10303-11].

3.2 Terms defined in ISO 10303-31

This standing document makes use of the following terms defined in ISO 10303-31:

3.2.1 **abstract test case (ATC):** a specification, encapsulating at least one test purpose, that provides the formal basis from which executable test cases are derived. It is independent of both the implementation and the values.

3.2.2 **abstract test method:** the description of how an implementation is to be tested, given at the appropriate level of abstraction to make the description independent of any particular implementation of testing tools or procedures, but with sufficient detail to enable these tools and procedures to be produced.

3.2.3 **basic tests:** limited tests performed to determine whether it is appropriate to perform thorough testing.

3.2.4 **conformance testing:** the testing of a candidate product for the existence of specific characteristics required by a standard in order to determine the extent to which that product is a conforming implementation.

3.2.5 **executable test case:** an instantiation of an abstract test case with values.

3.2.6 **fail (verdict):** a test verdict given when the observed test outcome demonstrates non-conformance with respect to either the test purpose or at least one of the conformance requirements in the relevant standard(s).

3.2.7 **Implementation Under Test (IUT):** that part of a product which is to be studied under testing, which should be an implementation of one or more characteristics of the standard(s) based on a given implementation method.

3.2.8 **inconclusive (verdict):** a test verdict given when the observed test outcome is such that neither a pass of a fail verdict can be given.

3.2.9 **pass (verdict):** a test verdict given when the observed test outcome gives evidence of conformance to the conformance requirement on which the test purpose is focused and is valid with respect to the relevant standard(s) and with respect to the PICS.

3.2.10 **postprocessor:** a software unit that translates product information from an independent public domain product data format to the internal format of a particular computer system.

3.2.11 **preprocessor:** a software unit that translates product information from the internal format of a particular computer system to an independent public domain product data format.

3.2.12 **Protocol Implementation eXtra Information for Testing (PIXIT):** a statement made by the client which contains or references all of the information (in addition to that given in the PICS) related to the IUT and its corresponding SUT, which will enable the testing laboratory to run an appropriate test suite against that IUT.

3.2.13 **test purpose:** a precise description of an objective which an abstract test case is designed to achieve.

3.2.14 **test realisers:** an organisation which takes responsibility for providing, in a form independent of the clients of a testing laboratory and their IUTs, a means of testing IUTs.

3.2.15 **(test) verdict:** a statement of "pass", "fail", or "inconclusive" concerning conformance of an IUT with respect to an executable test case and the abstract test case from which it was derived.

3.2.16 **verdict criteria:** information defined within an abstract test case which enables the testing laboratory to assign a verdict.

3.3 Other definitions

For the purposes of this standing document, the following definition applies:

3.3.1 **application element(AE):** an application object, attribute, or assertion defining the information requirements in clause 4 of an AP.

3.3.2 **coverage:** the percentage of all possible test purposes that are satisfied by input data specifications from at least one abstract test case.

3.3.3 **expected output specification (EOS):** a presentation of the output from a confroming implementation to be expected as the result of a conformance test.

3.3.4 **minimal entity set:** the set of AIM entity instances which shall be present in any instantiated model of a given AP.

3.3.5 **minimal object set:** the set of instances of application objects which shall be present in any instantiated model of a given AP.

3.4 Abbreviations

For the purposes of this SC4 standing document, the following abbreviations apply:

AE application element

AIM application interpreted model

AP application protocol

ARM application reference model

ATC abstract test case

ATS abstract test suite

EOS expected output specification

IP informal proposition

IUT implementation under test

SDAI standard data access interface

4 Overview

This document defines the form and content of an abstract test suite. It provides guidelines to an approach for developing abstract test suites for ISO 10303 conformance testing. These guidelines are based on the requirements set forth in the 30 series parts on conformance testing:

— ISO 10303-31 provides the overview of the concepts of conformance testing. High-level considerations specified there guide the entire conformance testing process and the development of test suites.

— ISO 10303-32 covers the relationships, requirements, and roles of test laboratories and their clients. ISO 10303-32 discusses the PIXIT, which depends in part on aspects of the abstract test suites.

— ISO 10303-33 defines how test realisers are to use the abstract test suite.

— ISO 10303-34 provides abstract test methods for conformance testing implementations of application protocols using the exchange structure defined by ISO 10303-21 or the standard data access interface ISO 10303-22.

NOTE 1 - The impact of ISO 10303-34 on the abstract test suites comes from the requirements of the abstract test methods.

— ISO 10303-35 provides abstract test methods for conformance testing implementations of the standard data access interface ISO 10303-22 (SDAI).

NOTE 2 - ISO 10303-35 has no impact on the current structure of abstract test suites.

Any characteristic, or combination thereof, called out in an International Standard may be the basis for conformance testing.

An abstract test suite for an AP is composed of a set of abstract test cases. Each abstract test case specifies input data to be provided to the implementation under test, along with the associated verdict criteria. Abstract test cases are so named because they define the necessary tests in a form that is applicable to all processors to be tested. At the time of actual testing the abstract test cases need to be converted into executable test cases, which are in a form that can be executed on the processor being tested. The following are needed to build an abstract test suite:

— the application protocol;

— a means of describing abstract test cases.

Testing an implementation of an ISO 10303 application protocol requires assessing capability in three areas: syntax, structure, and semantics. Syntax and structure analysis only apply to preprocessor testing. Semantic analysis applies to both preprocessor and postprocessor testing. Syntax analysis involves checking that the output exchange structure of a preprocessor satisfies all the requirements of the applicable implementation method(s). Structure analysis ensures that the data model represented in a preprocessor output exchange structure satisfies all structural requirements of the AIM EXPRESS long form of that AP. Semantic analysis verifies that the semantics supported by the application protocol of

interest are accurately conveyed in the observed output of both preprocessors and postprocessors. Numerical analysis is a part of semantic analysis that involves checking specific numeric output values against the corresponding input data values for equality within a specified numerical accuracy for testing.

Figure 1 presents an overview of the preprocessor and postprocessor conformance test process using the implementation method defined in ISO 10303-21 or ISO 10303-22. The basic approach of the test process is that an Implementation Under Test (IUT) is supplied an instance of the information model defined by the relevant conformance class of the application protocol in one format and is expected to translate it into another format. For a preprocessor, the specific format of the input is not defined by ISO 10303. For conformance testing purposes, it may be defined to be some form of human-readable text and graphics ("hardcopy") that represents an instance of the information requirements given in clause 4 of an AP. The preprocessor output format is an ISO 10303-21 exchange structure or a series of ISO 10303-22 calls that provide a model description.

For a postprocessor, the situation is reversed. The input format is an ISO 10303-21 exchange structure or series of ISO 10303-22 calls for model construction, while the specific syntax and structure of the output is undefined by ISO 10303. The semantics of the output, however, must match those encoded in the input as defined by the AP. For conformance testing purposes, postprocessor output is a series of responses to human-interpretable queries about the semantics contained in the input model instance. Though a postprocessor may provide hardcopy output, this is not assumed or required.

The overall process for creating the abstract test suite is:

a) Generate the appropriate set of test purposes from an AP that describe the set of testing objectives identified as important and relevant by AP domain experts.

b) plan the abstract test cases and map the test purposes to each.

c) Generate test case input data specifications and verdict criteria for preprocessor implementations and postprocessor implementations in parallel, both driven by the same input application semantics. Test purposes derived from the information requirements describe the testing objectives to be covered by input data associated with test cases for ISO 10303 preprocessor and postprocessor implementations. Test purposes derived from the AIM, together with those from the information requirements, describe the testing objectives to be covered by input data associated with test cases for ISO 10303 postprocessor implementations.

d) Construct a single abstract test case for preprocessors and postprocessors, using the parallel sets of input data specifications and verdict criteria.

e) Add the abstract test case to the abstract test suite.

f) Assess how well the test suite covers the test purposes as a whole and by each conformance class. Coverage is discussed in 4.2.

g) Loop back to step b until test purpose coverage is adequate.

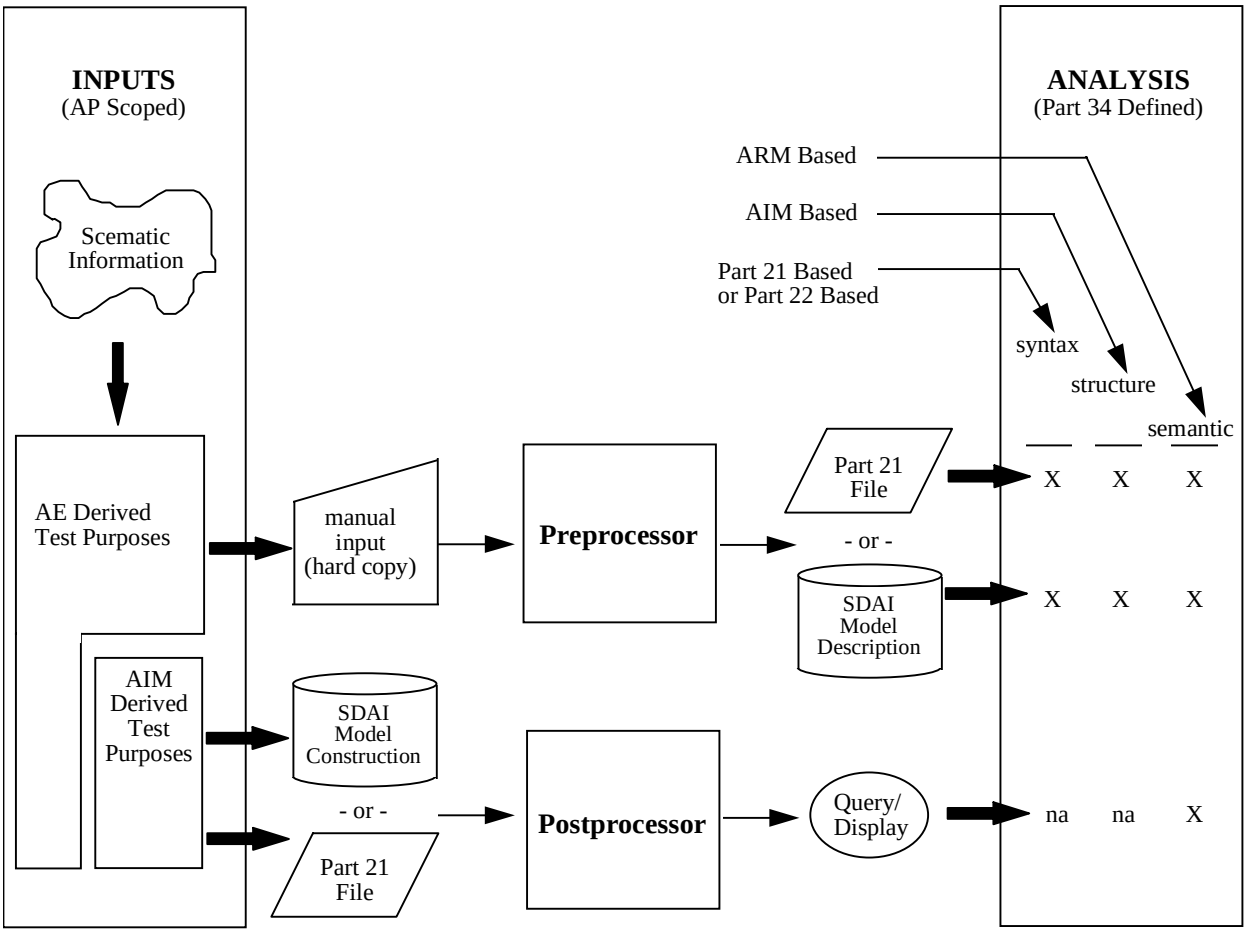

Figure 1 - Conformance test process

ATS development should proceed in parallel with the development of the relevant AP. That is:

a) Development of the application element (AE) test purposes should be undertaken before the information requirements of the AP are submitted for any formal reviews.

b) The full set of test purposes and at least some example abstract test cases should be developed before the AP is released as a committee draft.

c) The initial draft of a complete ATS should be ready at the same time or before the AP reaches final draft international standard.

The following subclauses describe fundamental concepts related to abstract test suites and their development.

4.1 Test purposes

Test purposes are the documented requirements of an ATS that specify the scope or range of the input data associated with the test cases in the test suite. Test purposes are derived by breaking down the general conformance requirements of an AP into a finite set of explicit statements representing the test objectives to be covered by the ATS. The fundamental conformance requirement present in all ISO 10303 APs is the preservation of the semantic content in the transferred product model. Application semantics must be encoded in an instantiation of the information model defined by an AP, and then be provided as input or produced as output by an implementation under test.

Test purposes are useful as a guide in the development of input data and verdict criteria for the abstract test cases that make up a test suite. An individual test purpose is a testing objective related to some aspect of an ISO 10303 application protocol. Test purposes typically describe application information requirements, data structure, constraints, or operations on instantiated models and model data. A test suite describes inputs to an implementation under test designed to cover the set of test purposes defined for the AP.

Test purposes derived from the information requirements describe the testing objectives to be covered by input data associated with test cases for preprocessor and postprocessor implementations. The information requirements are defined as AEs in clause 4 of an AP. These application semantics are encoded as input data for both preprocessor and postprocessor implementations. Application semantics as defined by the AEs are presented directly as input test case data to a preprocessor implementation. Each input specification covers some subset of the total set of AE test purposes generated for the abstract test suite.

Test purposes derived from the AIM, together with AE derived test purposes, describe the testing objectives to be covered by input data associated with test cases for postprocessor implementations. Application semantics as defined in clause 4 of the AP correspond to the AE test purposes. These semantics are driven through the AP mapping table to produce an instantiation of the AIM EXPRESS information model. This instantiation is presented as input test case data to a postprocessor implementation. The input specification may be presented as an ISO 10303-21 exchange structure or a series of ISO 10303-22 calls. This input specification covers both the AE test purposes and their corresponding AIM-derived test purposes, describing both the input semantics and the input structure.

Test purposes may also arise from requirements implicit in the AE of an AP. ATS developers shall decide if such a requirement should result in an explicit test purpose. Other categories of test purposes may be created by ATS developers. The creation of external reference test purposes are caused by the inclusion of other standards by an AP. Test purposes may result from specific requirements that may be present in Annex C of an AP (entitled “Implementation method specific requirements”). In all cases that such test purposes are present, the abstract test suite shall contain input specifications that adequately cover these test purposes.

4.2 Coverage

Coverage is a measure used by abstract test suite developers to ensure that an abstract test suite is adequate in scope and extent. Conformance testing has an inherent and unavoidable limitation: no theoretical or practical approach to testing or any other means of verification or validation can guarantee that an implementation is fault free. Since exhaustive testing is impossible, the abstract test suite developers shall establish when an abstract test suite is good enough. The set of test purposes and the

related notion of coverage establish the requirements for an adequate abstract test suite. The test purposes establish the requirements to be covered by the input specifications and verdict criteria of the set of abstract test cases. Coverage measures the degree to which the abstract test case input specifications represent the semantics and structures described by the set of test purposes.

Two conflicting needs must be balanced in the development of an abstract test suite: the need for testing an implementation as thoroughly as possible and the need to keep the cost of testing within practical bounds. The abstract test suite developers are responsible for determining the number and size of the abstract test cases and their individual scopes such that the coverage is sufficient.

4.3 Verdict criteria

Verdict criteria are the documented requirements which are used by the test laboratory to ascertain whether a test purpose has been satisfactorily met by the IUT within the context of a given abstract test case. Each verdict criterion is explicitly associated with one or more test purposes. A complete abstract test suite shall have at least one verdict criterion associated with every test purpose listed in clause 4 of the abstract test suite. Verdict criteria are developed and documented within the ATS to explicitly specify the acceptable results of a test. Verdict criteria are documented as either general, derivable, or specific verdict criteria. A general verdict criterion verifies that its associated test purpose(s) is(are) satisfied across multiple abstract test cases (see 6.6). Derivable and specific verdict criteria verify that their associated test purposes are satisfied in a particular abstract test case.

Verdict criteria are only applicable to the requirements specified within the AP and its normative references. Verdict criteria will generally be applicable to one of the three types of analysis:

— syntax analysis - checking an output exchange structure from a preprocessor to determine whether it meets the requirements of ISO 10303-21 or ISO 10303-22 as driven by a single general syntax verdict criterion;

— structure analysis - checking an output exchange structure from a preprocessor to determine whether it meets the structural requirements in the AIM EXPRESS of the relevant AP, as driven by the general structural verdict;

— semantic analysis - checking proper semantic content in the observable outputs of the IUT, for both preprocessors and postprocessors. as driven by general verdict criteria and any applicable derivable and specific verdict criteria.

4.4 Minimal object and entity sets and basic tests

Every AP has some limited set of information requirements that are required in every semantic model. Likewise, an AP's AIM EXPRESS schema (long form) requires a minimal entity set of AIM entity instances which have to be present in every instantiated model for a given AP.

The minimal object and entity sets can be used to construct basic tests as defined in ISO 10303-31. In such case the minimal object and entity sets will be defined for each conformance class. Every abstract test case shall include at least the minimal object set in the input specifications. In addition, the postprocessor input specifications shall include at least the minimal entity set.

Every object in the minimal object set and every entity in the minimal entity set need not be explicitly checked via verdict criteria in every abstract test case. The minimal object and entity sets for an AP may be empty when an AP has two or more non-intersecting conformance classes. Basic tests are one or more abstract test cases which shall be executed first. The purpose of the basic tests is to ascertain whether the implementation demonstrates a sufficient level of conformance to warrant the execution of the full set of tests. The abstract test case(s) that have an input specification and associated verdict criteria focused on the "minimal entity set" shall be part of the basic tests.

5 Sources for abstract test suite development

The following paragraphs describe how the different parts of ISO 10303 serve as sources for creating the abstract test suite.

5.1 Application protocol

The conformance requirements clause of an AP is the basis for test purpose generation, the definition of conformance classes, and the implicit requirements of the AP and its normative references. The abstract test suite defines and addresses a set of test purposes developed from the AP. The development of test purposes from an AP is described in clause 6.5 and annexes A, B, and C of this document.

5.2 Implementation methods

The implementation methods defined in the 20 series parts affect the abstract test suite. Both the exchange structure implementation method (ISO 10303-21) and the SDAI implementation method (ISO 10303-22) require separate preprocessor and postprocessor conformance testing.

The requirements of ISO 10303-21 and ISO 10303-22 on the characteristics of an exchange structure or SDAI interface itself also affect conformance testing. Test purposes and perhaps abstract test cases beyond those required by the AP itself are likely to result from the use of ISO 10303-21 exchange structures. The same is true for ISO 10303-22 implementations.

EXAMPLE 1 - ISO 10303-21 requirements outside the AP include user-defined entities, scope constructs, and exchange structure header entities.

Regardless of the differences between them, the conformance testing requirements of both ISO 10303-21 and ISO 10303-22 deal with the syntax of the product data exchange and, as such, can be handled in the general test purposes and verdict criteria.

5.3 Application interpreted constructs

AICs are still an emerging concept in ISO 10303. Although they are likely to be associated with abstract test suite components, discussion of these relationships is excluded.

5.4 Other requirements

The abstract test suite can also be affected by normative references to standards outside of ISO 10303. Conformance requirements resulting from referenced standards may need to be incorporated into the test purposes and abstract test cases.

EXAMPLE 2 - AP 201 and AP 202 both reference fonts that are defined in other ISO standards. AP 201 references ISO 3098-1:1974, Technical drawings - Lettering - Part 1: Currently used characters.

6 Abstract test suite documentation and development

The abstract test suite for the AP defined by ISO 10303-2nn is contained in ISO 10303-3nn, where nn is the same in both numbers

EXAMPLE 3 - For AP 207, the abstract test suite will be contained in part 307.

This clause defines the overall structure and content of the part 300 series (the abstract test suites). The clauses for a 300 series part are:

— Scope

— Normative references

— Definitions

— Test purposes

— General test purposes and verdict criteria

— Abstract test cases

This is based on the requirements for ISO standards that guide the development of all ISO 10303 parts (ISO Directives part 3 and the SC4 Supplementary Directives) and the requirements put forth in ISO 10303-33. For templates to be used in documenting 300 series parts, including some recommended text, see annexes E and F.

6.1 Administrative information

The following administrative information shall be included in the abstract test suite.

— the abstract test suite name and date of publication will appear in the normal manner of an ISO standard;

— the International Standard that specifies the application protocol to which the abstract test suite applies belongs in the Foreword and the list of references in clause 2 of the abstract test suite;

— a foreword, introduction, and title.

See E.2-E.5 for further information.

6.2 Documentation of scope

According to ISO requirements, clause 1 of an abstract test suite shall describe the scope of that abstract test suite. The scope clause shall include the following piece of administrative information :

— an identification of the technical corrections to the International Standard that specifies the application protocol that have been taken into account in the abstract test suite.

See E.6 for a template for the abstract test suite scope clause.

6.3 Documentation of normative references

All normative references shall be listed in clause 2 of the abstract test suite. The minimal required set of normative references for an abstract test suite are listed in E.6. Other references may be required, depending on the AP. For example, ISO 10303-201 refers to four standards in addition to ISO 10303. Those references will almost certainly be required in the abstract test suite, as conformance to their requirements needs to be addressed as well. In general, any standard that results in test purposes shall be included in the reference list.

See E.7 for a template for the abstract test suite references clause.

6.4 Documentation of definitions and abbreviations

As required by the ISO TC184/SC4 Supplementary directives, clause 3 of the abstract test suite shall include definitions of all concepts necessary to understand the document, either by reference to another part or by an explicit definition. The Introduction and Scope clauses are the most likely portions of the document in which terms (not definitions) in referenced documents will appear.

Clause 3 of the abstract test suite shall also provide a list of symbols and abbreviations used in the abstract test suite.

See E.8 for a template for the abstract test suite definitions and abbreviations clause.

6.5 Documentation of test purposes

Clause 4 of the abstract test suite contains a list of the test purposes covered by the abstract test suite. Test purposes are developed based on the conformance requirements in the AP and the implicit requirements of the rest of the AP and its normative references. The syntax for documenting test purposes is given in annex A. The test purposes shall be constructed using the specifications contained in this document (see Annexes B, and C). Test purposes are documented with a reference to the abstract test case(s) in which the test purpose has a verdict applied to it.

6.5.1 Application elements

AE test purposes are developed from the application objects and application assertions that represent the normative information requirements of the AP. AE test purposes represent the semantics of the domain

of an AP. A comprehensive test suite satisfies all AE test purposes to insure adequate coverage of the semantic scope and context of the application domain.

Application objects are atomic elements that embody a unique application concept and contain attributes specifying the data elements of the object. AE test purposes developed from application objects describe the atomic semantic elements of the domain of an AP. The description of the application objects may also specify valid values for those objects. If so, the valid values need to be included in the test purposes.

Application assertions specify all relationships among application objects, the cardinality of the relationships, and the rules for the integrity and validity of the application objects. AE test purposes developed from application assertions represent these constraints on the objects in the domain and scope of an AP.

The specific details of generating the AE test purposes are presented in annex B. The interpretation of each test purpose derived from the information requirements is given as follows: the IUT shall preserve the semantic associated with the unique application element from which the test purpose was derived. This implies that the semantics of the application element are preserved by the IUT between the input and output of a test according to the reference path specified in the mapping table of the AP.

As the mapping table defines a relationship between application elements and their corresponding AIM constructs, so the AE derived test purposes have a correspondence to the AIM test purposes derived from their related AIM constructs. These related AIM test purposes will be among those covered by the associated “mirror” postprocessor input specification within a single test case.

AE test purposes are directly useful in developing preprocessor input specifications, the “hardcopy” semantic input corresponding to the AP information requirements. AE test purposes are also important for generating specific verdict criteria for the outputs of both preprocessors and postprocessors.

Test purposes developed from the AEs are grouped according to the application objects defined in the information requirements. Each application object has an associated test purpose. Test purposes derived from the attributes of an application object are grouped immediately beneath the object test purpose. All assertion test purposes are grouped with the application object that appears first in the assertion test purpose name (forward assertions with the "from" object and inverse assertions with the "to" object). Assertion test purposes appear immediately after the attribute test purposes. All AE derived test purposes shall be addressed by the test case data associated with the test suite. See annex B for details on AE test purpose development.

6.5.2 Application interpreted model

AIM test purposes are developed from the long form of the AIM (Annex A of an AP). AIM test purposes represent the structure of the EXPRESS information model. A comprehensive abstract test suite ensures adequate test coverage of the structure of an AP’s EXPRESS information model by satisfying all AIM derived test purposes.

Each entity which can be instantiated has at least one test purpose associated with it. The details of generating the AIM test purposes are presented in annex C. Test purposes developed from the AIM are grouped according to the EXPRESS entities defined in the AIM.

The interpretation of each test purpose derived from the AIM EXPRESS schema is given as follows: the postprocessor shall accept the input in accordance with the AIM EXPRESS structure corresponding to

this test purpose. This implies that the semantics of the application element represented by the AIM element are preserved by the IUT between the input and output of a test according to the reference path specified in the mapping table of the AP. This also implies no violations of any constraints (e.g. where rules or global rules) that apply to the AIM element.

The basis for AIM test purposes are two high level conformance requirements expected to be present in all APs. For an AP:

— Only those constructs specified in the AIM shall be produced or accepted by a preprocessor or postprocessor implementation of the AP.[1]

NOTE 4 - Preprocessor implementation may produce user defined entities, but they are only checked for conformance to ISO 10303-21 syntax.

— All entities, types, and their associated constraints identified in a particular conformance class shall be supported by postprocessor implementations of the AP. Treatment of options and default values shall conform to the AIM.

These conformance requirements result in the following implicit global test purpose for an AIM:

— An instance of each entity in the AIM exists and all its attributes have values of the correct type which do not contravene any local rules, global rules, or informal propositions.

This implicit global test purpose applies to all entities defined in an AP. While it is globally true, this implicit test purpose provides no specific guidance in the testing of a particular data instance or a specific model. The generation of a set of specific, explicit AIM test purposes based on the global implicit AIM test purpose and the structure of the AIM provides this guidance. Explicit AIM test purposes benefit conformance testers and conformance test suite developers whose task is to create a finite and effective set of abstract test cases and assess the coverage of the input data associated with the abstract test cases in the test suite with respect to the test purposes.

AIM test purposes are particularly useful in developing test case data for postprocessor implementations which physically accept inputs in an ISO 10303 form (i.e., ISO 10303-21 or ISO 10303-22) corresponding to the structure of the EXPRESS information model. The structure of this postprocessor test data shall exercise the normative Application Interpreted Model (clause 5 of an AP) - represented by the AIM test purposes. AIM test purposes may also be useful for generating verdict criteria for the output of a preprocessor. See annex D for more information on verdict criteria and their development.

The structure of an EXPRESS information model is defined by the data types, relationships, and constraints of the corresponding AIM. Explicit AIM test purposes reflect the structure of EXPRESS entities, their attributes, and the constraints and relationships defined between them.

The specification of actual data values in a test purpose goes beyond the testing of the structure of data types and relationships. In general, data for individual attributes are assumed to be specific selections from a range of possible values. For some data types the number of possible specific values is infinite (e.g. REAL, STRING). For other data types, the number of specific values is potentially very large (e.g.

---

[1] From Guidelines for the development and approval of STEP application protocols, Version 1.1, November 1993, p 40.

ENUMERATION, SELECT). Others have a finite and relatively small number of possible values (e.g. BOOLEAN, LOGICAL).

Since testing all possible instantiated models is impossible, reasonable heuristics shall be adopted to guide the selection of representative values in the actual test case data. These heuristics are applied during the generation of the AIM test purposes to further restrict the subset of all possible information models addressed by an abstract test suite. They focus conformance testing on a useful set of representative test data values that are valid for the application domain and worth the cost of testing.

The AIM is developed through the use of integrated and application resources (the ISO 10303-40 and -100 series parts) in a process called interpretation. As a result, some of the AIM entities, attributes, and rules may not actually be of primary interest or value within a particular AP's domain, but are included in the AIM schema due to the ISO 10303 interpretation and integration process. Nonetheless, the algorithmic approach for developing AIM test purposes given in Annex C will generate test purposes for the entire AIM schema. AP domain expertise and judgment is required to identify any elements included in the AIM due to interpretation but not of primary interest within the application domain of the AP. It is recommended that test purposes derived from these ancillary AIM elements be excluded from the set of test purposes to be covered and documented in annex C of the ATS.

EXAMPLES

4 - If an AP does not address presentation, then an attribute for color for a geometric entity such as a line may be of no use, even though it was necessarily included in the adopted piece of AIM from a resource part.

5 - If an AIM entity attribute is defined as being “for information only”, then the test purpose that addresses that attribute is a likely candidate for removal. An example is the attribute b_spline_curve.self_intersect. This attribute is a flag that indicates whether the given b-spline curve intersects itself. The attribute is listed as being for information only so that if there is a discrepancy between the curve data and this flag, the curve data takes precedence.

6 - When units of measurement are brought in from ISO 10303-41, the entire set of units comes as a single group. In any particular AP, such as AP203, many of those measures such as luminous_intensity_measure, are likely to be unused. Test purposes associated with such unused units shall be removed.

See annex C for detail on AIM test purpose development.

6.5.3 Other test purposes

Other test purposes may be appropriate as a result of conformance requirements arising from other aspects of the AP or from other aspects of ISO 10303. These other test purposes may be derived from requirements that are implicit within the scope of an AP, from specific other parts of ISO 10303, or from other related standards. All other test purposes have at least one associated general or specific verdict criterion.

6.5.3.1 Domain test purposes

Domain test purposes arise from requirements that are implicit in the AE of an AP but are important enough to make into an explicit test purpose. These requirements may exist at a finer level of detail than is explicitly provided in the AEs of the AP.

EXAMPLE 7 - ISO 10303-201 requires that an **annotation_subfigure** have scaling, but does not give guidance as to what the scaling should be. Accordingly, the following domain test purposes have been added to ISO 10303-301 to ensure that scaling is correctly handled:

— annotation_subfigure with scaling 1:1;

— annotation_subfigure with scaling 1:n;

— annotation_subfigure with scaling n:1.

Domain test purposes may also describe explicit combinations of elements in particular configurations that are characteristic within the AP application domain. Domain test purposes are optional at the discretion of the ATS developer using AP domain expertise and judgment to determine if they are necessary . Domain test purposes shall not add requirements beyond those in the associated AP.

6.5.3.2 Implementation methods

The implementation methods described in the part 2x series may be the basis for specific test purposes. ISO 10303-21 exchange structure header requirements, for example, are a source of test purposes for abstract test suites designed to test implementations that use the exchange structure. An AP may have requirements specific to an implementation method as expressed in its normative annex C, titled "Implementation method specific requirements." Implementation method specific test purposes are needed to determine if these requirements have been met.

An implementation method may have more than one interface that requires testing. For example, the exchange structure implementation methods of ISO 10303-21 and ISO 10303-22 must be broken down by preprocessor versus postprocessor. Test purposes need to be specific to the interface for an implementation method. For a given test purpose, different abstract test cases may be needed for preprocessor testing as opposed to postprocessor testing.

6.5.3.3 Normative references to other standards

Test purposes are not only derived from the AP itself. Standards represented as normative references in the AP may contribute additional test purposes. Based on their knowledge of the domain, the abstract test suite developers shall determine whether the content of the referenced standard places additional conformance requirements on implementations of the AP. Referenced standards may include standards outside of ISO 10303.

EXAMPLE 8 - The normative references clause of ISO 10303-201 cites four standards in addition to ISO 10303 which, through reference, contain provisions to ISO 10303-201. One of the four standards is ISO 3098-1: 1974 — Technical Drawings - Lettering - Part 1: Currently used characters. In ISO 10303-201 5.2.2.20, the **draughting_pre_defined_text_font** states that "the information on pre-defined text fonts which shall be supported by all implementations of this part of ISO 10303 is given in ISO 3098-1." This requirement suggests that one or more test purposes address the referenced, pre-defined font.

The test purposes shall be organized by the clause structure described in E.8. Brief text at the beginning of the test purposes clause that guides the reader through that structure is highly recommended.

Individual test purposes shall be identified with a unique identifier made up of a combination of a character string plus an integer number with no separator between them. Character strings shall

represent the source of the test purpose (e.g., ae , aim). Text shall be included that explains the meaning of the character strings and any categorisation used.

EXAMPLE 9 - A test purpose may look like “ae056 Drawing (see 6.4 and 6.8)”. “ae” indicates the source of the test purpose, which combined with the number 056 provides a unique identifier in the abstract test suite. “Drawing” is the text of the test purpose, and “(see 6.4, 6.8)” indicates that the test purpose is associated with at least one verdict criterion in the abstract test cases described in clause 6.4 and 6.8 respectively, of the abstract test suite.

See E.8 for a template for the abstract test suite test purposes clause.

6.5.4 Test purpose development

Test purposes are derived by breaking down the general conformance requirements of ISO 10303 into a finite set of explicit statements representing the test objectives to be covered by the ATS. Conformance requirements arise from specific sections of an AP as well as from other parts of ISO 10303. The challenge of test purpose development is to produce a set of test purposes that is finite and manageable in size, but is also comprehensive and explicit enough to avoid mis-interpretation of the Standard during the development of test cases and subsequent conformance testing.

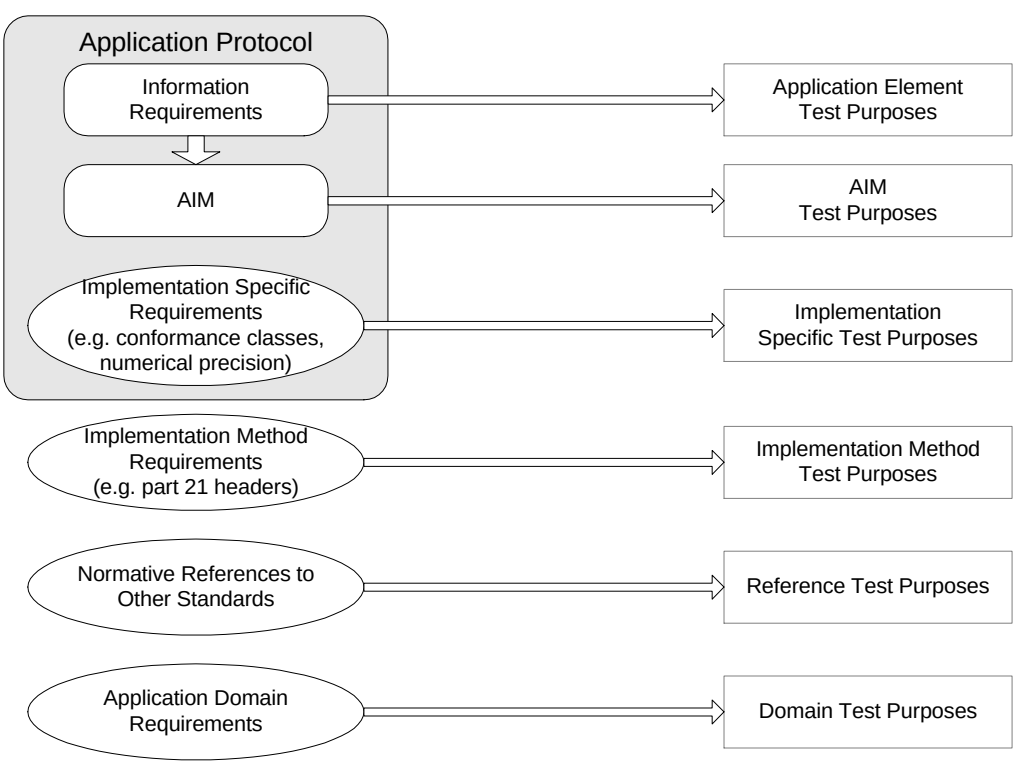

Figure 2 - Test purpose sources

Useful test purposes for an AP are developed from various sources (see figure 2). There may be other sources of test purposes not covered here. Examples of such sources may be AICs or normative annexes

of APs. The methods and sources listed here for generating test purposes should therefore be considered only the means for generating the kinds of test purposes likely to be common among all abstract test suites. Annexes B and C describe test purpose development in detail.

6.6 Documentation of general test purposes and verdict criteria

To avoid unnecessary repetition and encourage clarity, test purposes and their associated verdict criteria that apply to every abstract test case, to all preprocessor test cases, or to all postprocessor test cases shall be given in clause 5 of the abstract test suite.

General test purposes shall be stated in natural language as requirements on an implementation of the STEP AP. General verdict criteria shall be stated as natural language statements that can be evaluated as PASS, FAIL, or INCONCLUSIVE and formalized as necessary to ensure they are unambiguous.

See E.9 for a template for the abstract test suite general test purposes and verdict criteria clause.

6.7 Documentation of abstract test cases

Clause 6 of the abstract test suite shall contain the abstract test cases that make up the abstract test suite.

The abstract test suite shall contain abstract test cases that apply to preprocessors and postprocessors. Preprocessor and postprocessor input specifications with the same input semantics are documented together in the same abstract test case.

Abstract test cases may be sequenced in any manner the developers deem useful. Abstract test cases may be grouped according to basic tests driven by the minimal entity set. Abstract test cases may also be ordered by AP conformance class. If abstract test cases apply to more than one conformance class, a table shall be included in annex A of the abstract test suite that identifies for each conformance class the set of abstract test cases that apply.

EXAMPLE 10 - If an abstract test suite has no conformance classes but has an identified set of basic tests, then the first abstract test cases in clause 6 would be the basic tests, followed by the other individual abstract test cases.

NOTE 3 - No subjective qualification of abstract test cases shall be documented within the abstract test suite. E.g., terms such as "real-world" or "synthetic" test cases, even though such distinctions may be made during the development of the abstract test cases.

Each abstract test case shall be documented as a separate subclause of the abstract test cases clause (normally clause 6). An abstract test case is made up of a set of elements, each identified by a keyword heading immediately preceded and followed by a blank line. The keyword headings are as follows:

— Abstract test case identifier

— Test case summary

— Preprocessor

  - Test purposes covered

- Input specification
- Verdict criteria

— Postprocessor

- Test purposes covered
- Input specification(s)
- Verdict criteria

The documentation requirements for these elements are provided in the following subclauses. See annex F for a template for an abstract test case.

6.7.1 Abstract test case identifier

An abstract test suite is made up of a set of abstract test cases. In order to identify an individual test case, it needs an identifier that is unique within the abstract test suite. Therefore, each abstract test case shall be documented as a separate subclause of the abstract test suite. The subclause number then serves as the unique abstract test case identifier.

With the subclause number serving as the identifier of an abstract test case, the subclause title serves as the title of the abstract test case. The title may be any useful and appropriate informal reference to the test case.

6.7.2 Test case summary

In order to assist users in understanding the scope of each abstract test case, a test case summary shall be included in each abstract test case. The purpose of the test case summary is to provide an explanation of the high-level goal of the abstract test suite. The test case summary shall be the first item in the abstract test case and shall be introduced by the keywords:

```
Test case summary:
```

Test case summaries shall be concise and are typically no more than a few sentences.

EXAMPLE 11 - An example test case summary is: “This test case checks the ability of an IUT to correctly instantiate an advance b-rep solid model. The minimal amount of configuration management information is included to produce a conforming model, but none of that data is checked in this test case.”

6.7.3 Preprocessor

Each abstract test case shall contain a subclause that contains the input specification and verdict criteria for preprocessor testing. This subclause shall be the first subclause under the abstract test case subclause and shall be titled "Preprocessor". The preprocessor subclause shall be made up of three sections, the input specification, constraints on values, and the specific verdict criteria. The test purposes covered by a preprocessor input specification are listed in the Id column of the input specification.

6.7.3.1 Test purposes covered

The list of test purposes covered is the first section of the preprocessor subclause introduced by the keywords:

`Test purposes covered:`

with a blank line before and after the title. This section is used to list or reference all the general, AE or other test purposes covered by this preprocessor input specification. The identifiers of the AE test purposes present in the input specification are listed in the Id column of the preprocessor input specification. The V column identifies which of those AE test purposes are assigned a verdict (covered) in this test case. Boilerplate text is used to reference those test purpose identifiers without the need to explicitly repeat them in this section. In addition to the AE test purposes any general or other test purposes covered by this preprocessor test case are listed in this section using the boilerplate text and format defined in F.1.1.

6.7.3.2 Preprocessor Input specification

The preprocessor input specification information shall be the second section of the preprocessor subclause and shall be introduced by the keywords:

`Input specification:`

For preprocessor test cases, the input specification is a formalized presentation of the information requirements. It is represented by a combination of a table and supporting graphic and text information (sometimes called “hardcopy”). A template for the preprocessor input specification table is provided in F.1.1. This annex also contains a complete description of the syntax and detailed rules for creating a correct table. The table is composed of five columns:

— Column 1: the identifier for the row of the column, presented as the character @ followed by the test purpose id. If there is more than one instance of an application object with the same test purpose id in the input specification, the identifier is appended with a “.” followed by a number to make the entire identifier unique;

— Column 2: a flag in each row marking whether the application element contained in that row is to be verdicted. An asterisk (*) shall be used to indicate the verdict criteria are derivable. A number or a number range is used to refer to specific verdict criteria documented in the specific verdict criteria section;

— Column 3: The AEs to be addressed by the abstract test case, that is, application objects, attributes, assertions, and categorisations. The format is intentionally similar to the AP mapping table, both for the naming of the individual AEs and in the way that they are grouped;

- For application objects, simply give the name of the application element. Use upper case for the first letter only;

- For application object attributes use the notation Application_object.attribute This is done instead of using just the attribute name to allow each row of the table to stand by itself. Also, it allows for the clear identification of attributes and their owning application objects, especially in cases of complex relationships between application objects;

- For assertions, use the notation Application_object to Application_object (role);

- For categorisations, use the notation Application_object (as Application_object).

— Column 4: Values to be used for the application elements. Values may be specific values, references to other application elements (using the @ identification for the row), or references to information outside the table, such as a figure or the postprocessor input specification. See annex F for more detail on the syntax of this column;

— Column 5: A flag for whether the value in column 4 is mandatory, suggested, or constrained. For each value provided, the value of this flag shall be M for mandatory values, S for suggested values, or Cn for constrained values, where n is a unique integer that refers to a listed constraint in the constraints on values section. Normally values are considered suggested (i.e. the test realiser is free to change them within the allowed rules of the standard). Values may be constrained if the abstract test suite developer identifies specific rationale for the constraint and specifies the allowed alterations on the value. Values are mandatory due to requirements of the standard (EXPRESS rules, mapping table requirements, or test purposes). See annex F for specific rules on using M, S, or Cn.

Additional information to remove ambiguity in an input specification could include:

— diagrams or other graphical representations that clearly describe requirements but do not contain any more information than required to unambiguously represent the semantic content;

EXAMPLE 12 - If the AP does not support the exchange of presentation information such as color, line fonts or dimensions, no graphical representation used in an abstract test case for that AP shall suggest a specific color or line font. Likewise, any dimensional information provided shall be clearly identified as for information only.

— AIM constructs supporting and clarifying the AE test purposes.

6.7.3.3 Preprocessor Constraints on values

This optional section is only included if any values in the input specification are constrained (denoted with a Cn in column 5 of the table). This section contains a list of each constraint (each starting a new line) which defines the nature of the constraint and what changes to the value are allowed. This section of the preprocessor test information subclause shall be introduced by the keywords:

```
Constraints on values:
```

6.7.3.4 Preprocessor verdict criteria

Verdict criteria are statements of the requirements that shall be satisfied for a PASS verdict to be assigned. They are assertions on the observable output of an implementation under test resulting from its execution of a test case. Specific verdict criteria defined in this section of the abstract test case shall be explicitly associated with the test purposes they address. The verdict criteria shall be the third section of the preprocessor test information subclause and shall be introduced by the keywords:

```
Verdict criteria:
```

The specific verdict criteria shall be expressed in natural language, formalized as necessary to provide a clear, concise statement. Preprocessor verdict criteria shall be phrased as statements that can be evaluated as PASS, FAIL, or INCONCLUSIVE. The development of verdict criteria is discussed in 6.7.5.

Annex D provides algorithms for deriving verdict criteria. Those specific verdict criteria derived from the algorithms of annex D are for the purposes of test case realisation, and shall not be documented in the abstract test suite. Only those additional specific verdict criteria developed beyond the methods described in annex D are to be documented in this section of the abstract test case. In addition, any general verdict criterion defined in clause 5 of the abstract test suite that applies to this test case shall be identified here by reference, according to the boilerplate text for this section provided in annex F.

In the case of a preprocessor test, the input of the test is described using the ARM terminology (i.e., application elements) of the AP to which the IUT claims conformance. Additional information may be needed if the application elements do not express all the information that is required for an unambiguous description of the preprocessor input.

EXAMPLE 13 - AP 203 does not specify the details of a B-Rep model in its ARM. Therefore the ATS developer may choose to provide the details of a B-Rep solid in the form of a reference to a figure showing a drawing within the input description.

For each ATC, the input of the preprocessor is described in the form of a table giving suggested or mandatory values for application elements and their attributes. Instances of application constructs that are provided in order to exercise the AE test purposes related to the ATC are marked by an asterisk. For these instances, the observed output of the IUT has to be investigated.

As long as the preprocessor input specification does not go beyond the detail of the AP's application elements, the mapping table given in clause 5 of the AP describes precisely what the corresponding output of a conforming implementation ought to be. In this case the verdict criteria can be derived directly from the mapping table and specific verdict criteria are not required. An algorithm to do this is presented in annex D. The task of exercising this algorithm in order to generate the derivable verdict criteria is not the responsibility of the ATS developer, but is part of the construction of an executable test suite as described in ISO 10303-33.

Specific verdict criteria for preprocessor testing need to be provided for the individual abstract test cases where the verdicted preprocessor input description provides more detail than the related AP's application elements. Such specific verdict criteria are required to ensure that a conformance testing laboratory using the abstract test suite will, in fact, cover all the aspects of the input specification when analyzing the IUT's output.

EXAMPLE 14 - In the case given in the example above, the verdict criteria derived from AP 203's mapping table will cover the test whether an advanced_brep_shape_representation is present at all. In order to capture the additional information provided by the drawing, the ATS may contain the following verdict criterion: "The geometry in the preprocessor output matches the specifications of the drawing given in figure x.".

6.7.4 Postprocessor

Each abstract test case shall contain at least one separate subclause documented for each associated postprocessor input specification. More than one postprocessor input specification may be required to cover the various acceptable input forms. This happens as a result of multiple ways in the AIM EXPRESS to encode a particular set of semantics expressed in the application elements.

EXAMPLE 15 - For the application object **date**, there are a number of legitimate ways to represent date in the AIM EXPRESS (e.g., calendar date, ordinal date). If the application semantic to be evaluated is **date**, then the various input specification variations of date shall be presented in the input specifications for postprocessors.

Each postprocessor subclause contains the AIM test purposes covered, a reference to the postprocessor input specification, and any specific verdict criteria required for testing. All postprocessor input specifications accompany the ATS in digital form on diskettes or other media. The postprocessor subclause(s) shall begin as the second subclause under the abstract test case. A single postprocessor subclause shall be titled "Postprocessor". If there are two or more postprocessor input specifications for an abstract test case, they shall be listed sequentially in separate subclauses, each titled "Postprocessor n" where n is a sequential integer which uniquely identifies the postprocessor input specification within the abstract test case.

Each postprocessor subclause is divided into separate sections for the test purposes covered, the postprocessor input specification reference, and the verdict criteria.

6.7.4.1 Test purposes covered

Each abstract test case may address one or more AIM test purposes in its input specifications and verdict criteria. Each abstract test case shall list the test purposes identifiers for all AIM test purposes that are assigned a verdict in each postprocessor input specification for the test case. The list of covered test purposes is the first section in each postprocessor subclause and shall be introduced by the keywords:

```
Test purposes covered:
```

6.7.4.2 Postprocessor Input specification

The postprocessor input specification information shall be introduced the second section of each postprocessor subclause. The actual postprocessor input specifications for all ATCs accompany the ATS in digital form on diskettes or other media. They are referenced in the ATC by the keywords:

```
Input specification:

See Annex C
```

The normative Annex C establishes the link between each ATC and its associated postprocessor input specifications on the accompanying digital media.

The input specification defines the concepts, constructs, and relationships to be tested without mandating specific data values, except where the test purpose itself mandates specific values. If suggested values are provided as part of an input specification, they shall not represent any actual person, company, or organization.

The postprocessor input specification shall be in ISO 10303-21 or a form from which a ISO 10303-21 exchange structure can be generated. Accordingly, the postprocessor input specification shall be provided in one of the following:

— ISO 10303-21;

— ISO 10303-12 (EXPRESS-I).

Other formal methods may come into existence, but at this time, these are all that are approved for describing postprocessor input specifications. If possible, abstract test suite developers should choose one method for use throughout the abstract test suite to help ensure consistency.

6.7.4.3 Postprocessor verdict criteria

As with the preprocessor, the postprocessor verdict criteria are statements of the requirements that shall be satisfied for a PASS verdict to be assigned. There is a single set of verdict criteria for all postprocessor input specifications in a given abstract test case. This is because the semantic content is the same in the different representations of the postprocessor input specification. The verdict criteria follow the postprocessor input specification section(s) and shall be introduced by the keywords:

```
Verdict criteria:
```

The specific verdict criteria shall be expressed in natural language, formalized as necessary to provide a clear, concise statement. Postprocessor verdict criteria shall be phrased as statements that can be evaluated as PASS, FAIL, or INCONCLUSIVE. The development of verdict criteria is discussed in 6.7.5. Only those additional specific verdict criteria developed beyond the methods described in annex D are to be documented in this section. Those specific verdict criteria derived from the algorithms of annex D are for the purposes of test case realisation, and shall not be documented in the abstract test suite. In addition, any general verdict criterion defined in clause 5 of the abstract test suite that applies to this test case shall be identified here by reference.

In the case of a postprocessor test, the input of the IUT is described using the AIM terminology of the AP to which the IUT claims conformance, i.e., a file encoded according to ISO 10303-21 or a sequence of calls according to ISO 10303-22.

The output of a postprocessor is dependent on the IUT, but the IUT is used in the context of an AP. The syntax and structure of the application elements of the AP are therefore suitable to describe the IUT's output in a system-independent manner. The verdict criteria use the terminology of the application elements to allow the verification of the IUT's behaviour.

For each ATC, the expected output of the postprocessor is described by the input specification of the associated preprocessor. This is presented in the form of a table giving suggested or mandatory values for application objects and their attributes. For the case of suggested values, the testing laboratory conducting the conformance test will assign fixed values during the creation of executable test cases and assign fixed values during the creation of executable test cases for postprocessor input. This is equivalent to having such a table which uses exclusively mandatory values. This finalized table is referred to as the Expected Output Specification (EOS). The EOS is a precise representation of the expected output from a conforming postprocessor. Therefore the verdict criteria can be directly derived from the EOS. An approach to translate the EOS to a series of verdict criteria is presented in annex D. The task of exercising the algorithm is left to the testing laboratory as part of the executable test suite generation as described in ISO 10303-33.

When the expected output is described using constructs other than application elements, additional specific verdict criteria are required because the success of the translation cannot be judged by testing the presence of application elements alone. These specific verdict criteria must also be included in ATS.

EXAMPLE 16 - In the case given in EXAMPLE 13 above, the verdict criteria derived from the expected output specification will cover the test whether an advanced_b_rep is present at all. In order to capture the additional information provided by the drawing, the ATS may contain the following verdict criterion: "The geometry data in the postprocessor output matches the specifications of the drawing given in figure x.".

6.7.4.4 Additional test case information

In an abstract test case, information beyond that given by the previous entries may optionally be included to ensure that an executable test case can be realized. If so, then that information shall be contained in a separate section of an abstract test case. This section can appear in any or all of three locations, in the common section at the beginning of the abstract test case, in the preprocessor subclause, and in the postprocessor subclause. This section shall be the last section of the part of the abstract test case in which it appears.

The kind of information that shall appear in the additional information section includes:

— Sequence of execution;

— Extra details on the test.

6.7.4.4.1 Sequence of execution

In the execution of a test case, the sequence of operations to be performed may be significant. If the order of creation, use, or modification of the components of the input specification may affect the test case outcome, then additional information identifying the required sequence shall be included in this section of the abstract test case. Such information may be presented through the use of natural language or by a means such as ISO 10303-22. If present, this information shall be introduced by the keywords:

`Execution sequence:`

Any information necessary to make clear the application of the execution sequence shall also be included.

6.7.4.4.2 Extra details on the test

If any further information is needed to describe the abstract test case, it shall be presented in a section called “Extra details.” If present, this information shall be the last section in the appropriate subclause of the abstract test case and shall be introduced by the keywords:

`Extra details:`

6.7.5 Abstract test case development

The value of the abstract test suite lies in the abstract test cases, each of which tests some functionality of the AP. The abstract test cases provide implementers and test laboratories with the information they need to test products for conformance.

The origin of a particular test case is not important. What is important is that each test case provides a realistic test of one or more test purposes. Therefore, abstract test cases should be based on realistic use of the information within the AP’s scope. One technique is to refer to the AAM for usage scenarios from which the ARM and AIM were developed. Any usage scenarios used for developing abstract test cases should be documented in an informative annex of the abstract test suite.

Developing abstract test cases is not a simple, well-defined process. Given that exhaustive testing is not possible, the application protocol developers and implementers must decide what to test and how

thoroughly to test it. In general, the greater the impact a proposed abstract test case provides in detecting and preventing product model data exchange errors, the more value it will provide as part of an abstract test suite. APs provide a formal structure with fully defined semantics for exchange. In spite of this, problems with exchange are certain to occur. If, in the standard or product development process, a test case is generated that is known to be useful in detecting or preventing a problem in product model data exchange, that test case should be included in the abstract test suite.

6.7.5.1 Structuring and creating abstract test cases

The following steps provide an approach to building test cases:

a) Identify those test purposes which may be combined into a single abstract test case. Use whatever approach makes sense in combining test purposes. Ensure that all test purposes that are assigned a verdict in a test case belong to the same conformance class.

EXAMPLE 17 - It is often sensible to identify the minimal object set and the minimal entity set, by definition required to exist in every abstract test case.

EXAMPLE 18 - It may be sensible to address those test purposes that share the same references in subclauses 4.2 and 4.3 of an AP in a single test case.

b) Develop the abstract test cases.

1) Develop a preprocessor input specification, identifying AE test purposes covered;

2) Develop corresponding postprocessor input specification(s), identifying AIM test purposes covered;

3) Develop specific verdict criteria, if needed.

c) Evaluate coverage and reiterate.

d) Group the abstract test cases according to conformance classes.

It may also be useful to work in the other direction for some test cases. That is, if a useful test case input specification has been identified through whatever means, then the test purposes covered by the specification(s) should be identified, specific verdict criteria developed as needed, and a complete abstract test case developed.

Each abstract test case addresses both preprocessor and postprocessor implementations. While the AE semantics apply to both, the input specifications for the two types are different. The abstract test case shall contain the input specifications and suitable verdict criteria for both preprocessor and postprocessor tests. Because a given application object may map to more than one AIM construct, an abstract test case may require more than one postprocessor input specification to be executed.

EXAMPLE 19 - Assume an AP has an application element b_spline_curve. Using the integrated resources (ISO 10303-42 in this case), that would map to eight types of b-spline curves. In an abstract test case designed to evaluate whether a processor could handle b-spline curves, the preprocessor input specification would include a b-spline curve. However, the equivalent postprocessor input specification would come in four forms, one for each of the types of b-spline curve allowed.

6.7.5.2 Input specifications

The input specification is a concise description of the input data requirements for the abstract test case. Each abstract test case will contain the specification for the input data to be used when developing and running an executable version of that abstract test case. The input specification defines the concepts and relationships to be tested without mandating specific data values, except where the test purpose itself mandates specific values. If suggested values are provided as part of an input specification, they shall not represent any actual person, company, or organization.

For preprocessor implementations the input specification is a formalized presentation of the information requirements, typically represented by a combination of text/tabular presentation and graphics (sometimes called “hardcopy”). See annex F for details on the preprocessor input specification.

6.7.5.3 Verdict criteria

Verdict criteria are developed by ATS developers to explicitly specify the acceptable results of a test. They are documented as either general, i.e. applicable to multiple test cases, or specific to a single test case. To verify that the requirements for a particular test purpose are fulfilled by an IUT, verdict criteria are established in the form of assertions on the IUT’s output that can be evaluated to PASS (test was successfully exercised) or FAIL (test showed a conformance violation by the IUT), or INCONCLUSIVE (it was not possible to determine a PASS or FAIL verdict).

Verdict criteria are used in the three basic kinds of testing analysis outlined in ISO 10303-34: syntax, structure, and semantic. Accordingly, the verdict criteria shall address all three of these areas:

— Syntax analysis can be addressed by a single general verdict criterion (which may be applicable across an entire ATS) which requires that all exchange structures produced by an ISO 10303-21 preprocessor meet the syntax requirements of ISO 10303-21. Likewise, a single general verdict criterion may require an ISO 10303-22 preprocessor to meet the appropriate syntax requirements of ISO 10303-22.

— Structure analysis can be addressed in the ATS by a single general verdict criterion that requires that the structure of the AIM is correctly maintained. For the purposes of test case realisation, an algorithm for the generation of an explicit series of specific structural verdict criteria derived from the AP mapping table is detailed in Annex D.1. Additional specific structural verdict criteria may also be developed to address specific aspects of AIM structure.

— Semantic analysis can be addressed in the ATS by a single general verdict criterion that requires that the semantics of the preprocessor input specification is correctly maintained. For the purposes of test case realisation, an approach for deriving specific semantic verdict criteria from the preprocessor input specification table is provided more detail in Annex D.2. Additional specific semantic verdict criteria may also be developed to help determine whether the information requirements in the input specifications are maintained.

A general verdict criterion is associated with a general test purpose, and verifies that its associated test purpose(s) is (are) satisfied across multiple abstract test cases. The following are example of general verdict criteria:

EXAMPLES

20 - gvc1 The semantics of the input model are preserved in the output of the IUT according to the reference paths specified in the mapping table defined in clause 5 of ISO 10303-203 (g1).

21 - gvc2 The output of a preprocessor conforms to the implementation method to which the IUT claims conformance (g2).

22 - gvc3 The instances in the output of a preprocessor are encoded according to the AIM EXPRESS long form and mapping table as defined in Annex A and clause 5 of ISO 10303-203 (g3).

23 - gvc4 The postprocessor accepts input data which is encoded according the implementation method to which the IUT claims conformance (g4).

24 - gvc5 A postprocessor accepts input data which is structured according to the AIM EXPRESS long form and mapping table as defined in Annex A and clause 5 of ISO 10303-203 (g5).

Derivable verdict criteria are developed corresponding to both the preprocessor and postprocessor intput specifications. Specific structural verdict criteria are derived for preprocessors from the application elements of the input specification table using the AP mapping table as described in annex D.1. Specific semantic verdict criteria are derived for both preprocessors and postprocessors from the realisation of the input specification table, the expected output specification, as described in annex D.2. Derivable verdict criteria shall not be documented in the abstract test suite, and shall be developed for the purposes of test case realisation.

Specific verdict criteria are not derivable from the AP. They verify the satisfaction of test purposes not directly relating to the reference path of the AP mapping table, and are are often associated with other test purposes or semantic validation at a level of detail beyond that provided by the AP application elements. These specific semantic verdict criteria are typically associated with a general semantic test purpose. The following are examples of specific verdict criteria:

EXAMPLES

25 - In a (pre)postprocessor test case, the general test purpose

g5 The output of an IUT shall contain all the semantics defined by the input model.

may have the associated specific verdict criterion

Geometry data in the postprocessor output preserves the semantics of the specifications given in figure 1

where figure 1 represents the geometry and topology of the input model.

26 - In a postprocessor test case, the general test purpose

g5 The output of an IUT shall contain all the semantics defined by the input model.

may have the associated specific verdict criterion

The diameter of the largest circle in this model is 2.5 cm.

Each specific verdict criterion shall be explicitly associated with each test purpose it addresses. A complete abstract test suite shall have at least one verdict criterion associated with every test purpose listed in the abstract test suite.

6.8 Abstract test suite annexes

The abstract test suite shall have at least four annexes, which provide information on:

— The relationship of the abstract test cases with the conformance classes of the AP (annex A);

— An ISO-required annex providing information object registration (annex B);[2)]

— The association of test case to electronic postprocessor input specification files (annex C);

— The list of test purposes that have been excluded and not have associated verdict criteria in any of the abstract test cases (annex D).

The abstract test suite part may include other normative and informative annexes as deemed appropriate by the test suite developers. In particular, it should include an informative annex delivered in electronic form that contains ISO 10303-21 exchange structures corresponding to the abstract test cases.

6.8.1 Documentation of conformance classes

The abstract test suite shall contain a normative annex A that defines which abstract test cases are appropriate for which conformance classes of the AP. If the AP has no documented conformance classes, then the annex shall include a statement to that effect.

See E.11 for more details on the documentation of this annex.

6.8.2 Documentation of postprocessor input specification file names

The postprocessor input specification for each test case is supplied electronically on magnetic media (floppy disk). The abstract test suite shall contain a normative annex C that lists the file names of the postprocessor input specifications supplied and associates them with the postprocessor subclause of each test case.

6.8.3 Documentation of excluded test purposes

The abstract test suite shall contain an informative annex that documents all test purposes that were generated from the AIM EXPRESS, but have been deliberately removed from the abstract test suite by application of domain expertise. An explanation as to why the test purposes were excluded from the abstract test suite shall also be provided in this annex. See 7.2 for more information on deliberate removal of test purposes.

See E.13 for more details on the documentation of this annex.

---

[2] This required annex is required of all ISO 10303 parts and is described in the Editing Committee's Supplementary Directives.

7 Test suite validation

The abstract test suite developers shall ensure that the abstract test suite is valid. Validation is required for both the abstract test suite as a whole and for each individual abstract test case. Validation is accomplished by the approach presented here. Validating the “correctness” of an abstract test suite and its constituent test cases will likely be achieved through three processes:

a) Verification of internal details through manual walk-through of selected samples of test cases;

b) Execution in real testing situations, and feedback of detected problems;

c) Use of reliable tools both in the generation and analysis of the tests.

Annex H provides checklists for the approval of abstract test cases and the overall abstract test suite.

7.1 Validating the abstract test suite

Abstract test suite validation involves three aspects. The first aspect of validity is established when the set of test purposes satisfies the implicit and explicit set of AP conformance requirements. The second aspect is established when the individual abstract test cases have been validated for their individual form and content. The third aspect of validity is coverage. Each abstract test case contributes to the overall coverage of the abstract test suite. Annex H provides a checklist for abstract test suite validation.

The following are the criteria that apply to evaluating a test suite as a whole. A good test suite shall:

— contain test cases that adequately cover the test purposes;

— contain a set of test purposes sufficient for the application domain;

— contain all the information needed for actual testing;

— not contain unnecessary redundancies;

— be practical for actual testing.

Validating an abstract test suite against its relevant AP is necessary to ensure that the entire abstract test suite serves the needs of the AP.

7.2 Validating abstract test cases

Individual abstract test cases shall be validated against the requirements specified in clause 9. Annex H provides a checklist for abstract test case validation. The following are the criteria for individual abstract test case validation:

— the test case identifier is present and correct;

— the test case summary is present and correct;

— the test purpose coverage is present and correct;

— the verdict criteria are present and correct;

— the input specification present is and correct.

Annex A
(normative)

Test purpose documentation syntax

This annex specifies a method for documentation of test purposes for inclusion within ISO 10303 Abstract test suites. This method is applicable to the formal specification of:

— test purposes derived from the application elements (AEs) of an application protocol;

— test purposes derived from the application interpreted model (AIM) of an application protocol;

— test purposes derived from domain requirements;

— test purposes derived from an implementation method;

— test purposes derived from standards that are referenced by an application protocol.

This method is also applicable to the identification of test purposes that are derived from other sources.

The following are outside the scope of this method:

— the formal specification of test purposes derived from any source other than AEs, AIM, domain, implementation method, or external reference;

— the specification of test purposes for any purpose other than the development of ISO 10303 abstract test suites;

— the requirements for software tools that may be used to develop, validate or process test purposes according to the syntax defined in this document.

A comprehensive set of test purposes is likely to be large and complex. Supplying the complete context for every test purpose will reduce the readability and clarity of the information. For this reason it is recommended that an explanation of how to interpret the test purposes is provided at the beginning of the test purposes clause. The following is a suggested version of that text:

Each test purpose statement identifies some specific element from the AEs or the AIM. Every test purpose statement implicitly requires that the identified element, as specified in the test purpose statement, will be correctly instantiated in the implementation under test in at least one test case within the test suite.

The following proposes the lexical elements of test purposes and the grammar rules which these elements obey. The notation used to present the syntax of test purposes is defined in clause 6 of ISO 10303-11.

The rules in the following subclauses specify the tokens used in test purposes.

A.1 Keywords

The following rules are used to represent the keywords of test purposes. Note that case is not significant except within string literals. Test purposes may be written using upper, lower, or mixed case letters. However, certain keywords are defined in all capital letters and shall be written as such in the actual test purposes.

0 ae = 'ae' .

1 aim = 'aim' .

2 and = 'AND' .

3 as = 'as' .

4 atc = 'atc' .

5 element = 'element' .

6 false = 'FALSE' .

7 many = 'many' .

8 not-present = 'not present' .

9 of = 'of' .

10 one = 'one' .

11 other = 'other' .

12 true = 'TRUE' .

13 unknown = 'UNKNOWN' .

14 with = 'with' .

15 zero = 'zero' .

A.2 Character classes

The following rules define the various classes of characters which are used in constructing the tokens in A.3.

16 digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' .

17 digits = digit { digit } .

18 letter = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z'.

19 sign = '+' | '-'.

A.3 Lexical elements

The following rules specify how certain combinations of characters are interpreted as lexical elements within the language.

20 integer-literal = digits .

21 logical-literal = FALSE | TRUE | UNKNOWN .

22 real-literal = digits '.' [digits] ['e' [sign] digits] .

23 simple-id = letter { letter | digit | '_' } .

24 special-char = '!' | '"' | '#' | '$' | '%' | '&' | '+' | ',' | '-' | '.' | '/' | ':' | ';' | '<' | '=' | '>' | '?' | '@' | '[' | '\' | ']' | '^' | '_' | '`' | '{' | '|' | '}' | '~' | '(' | ')' | '*' .

25 string-literal = \q { (\q \q) | letter | digit | special-char | \s | \o } \q .

26 strict-literal = \q { letter | \s | \o } \q .

A.4 Grammar rules

The following rules specify how the previous lexical elements may be combined into constructs for test purposes. The primary syntax rules for test purposes are 30, 31, and 42.

27 aggr-spec = of \s (zero | one | many | integer-literal) \s element[s] .

28 aim-attribute-spec = attr-aggr | attr-value | attr-optional .

29 aim-test-purpose = aim test-purpose-id \s object-name \s [aim-attribute-spec | relationship-spec | categorisation-spec] .

30 ae-attribute-spec = simple-attr | attr-aggr | attr-value | attr-optional .

31 ae-test-purpose = ae test-purpose-id \s object-name \s [ae-attribute-spec | relationship-spec | categorisation-spec] .

32 attr-aggr = simple-attr \s '=' \s aggr-spec .

33 attr-optional = simple-attr not-present .

34 attr-value = simple-attr \s '=' \s attribute-value .

35 attribute-name = simple-id .

36 attribute-value = literal | enumeration-reference .

37 categorisation-spec = as simple-categorisation | complex-categorisation .

```
38  complex-categorisation = '(' object-name AND object-name { AND object-name } ')' .
39  enumeration-reference = simple-id .
40  literal = integer-literal | logical-literal | real-literal | string-literal .
41  object-name = simple-id .
42  other-test-purpose = other test-purpose-id string-literal .
43  relationship-spec = role-description ( zero | one | many ) object-name .
44  role-description = { letter | \s | \o } .
45  simple-attr = with attribute-name .
46  simple-categorisation = object-name .
47  syntax = test-purpose {test-purpose} .
48  test-purpose = ae-test-purpose | aim-test-purpose | other-test-purpose .
49  test-purpose-id = digits .
```

Annex B
(normative)

AE test purpose development

Application element (AE) test purposes are derived from the normative information requirements defined in an application protocol. This annex describes a process to generate an explicit, comprehensive list of AE test purposes from the information requirements of clause 4 of the associated AP.

The interpretation of each test purpose derived from the information requirements is given in the following statement:

> Correctly instantiate in the implementation under test the semantic associated with the unique application concept corresponding to *(insert test purpose here)* in at least one test case within the test suite.

The *Supplementary directives for the drafting and presentation of ISO 10303* gives specific guidance on the documentation of these information requirements. It states that the information requirements are specified as a set of units of functionality, application objects, and application assertions. An application object is an atomic element that embodies a unique application concept and contains attributes that specify the data elements that the object is comprised of. The assertions pertain to individual application objects and to relationships between application objects. The application objects and application assertions are documented in the language of the application, but mixed with the terminology of data modeling.

AE test purposes are derived from the following aspects of the application reference model:

- application objects;
- application objects with categorisations (subtypes);
- application objects with mandatory attributes;
- application objects with aggregated attributes;
- application objects with specific attribute values;
- application objects with optional attributes;
- application assertions describing relationships between application objects.

This method results in the generation of at least one AE test purpose for each normative Information Requirement in clause 4 of the AP. A general term application element specifies either an application object, a specific attribute defined on an application object, or an application assertion. Every application element corresponds to at least one AE test purpose. AEs shall be documented with the same

convention used in clause 4 of the application protocol. The first letter of the first word of an application object name is capitalized, no letters of the attribute names are capitalized.

B.1 Test purposes derived from application objects

Every application object defined in the application protocol results in at least one test purpose for that application object.

B.1.1 Simple application objects

Application objects are defined in the subclauses of 4.2 in an AP. A simple application object is given in the AP information requirements without any defined categorisations. Each simple application object results in one explicit test purpose corresponding to an instance of that object created as itself. Each subclause of 4.2 of the AP that defines a simple application object results in one test purpose where the test purpose consists of the keyword “ae”, the test-purpose-id, and the application object name.

EXAMPLE 27 - In ISO 10303-201

4.2.30 Drawing

results in the test purpose:

ae001 Drawing

B.1.2 Categorisations

In the subclauses of 4.2 of the AP, an application object may define categorisations using one of the following:

Each <application object name> may be a(n) <subtype application object name 1>, or a(n) <subtype application object name 2>, ... or a(n) <subtype application object name n>.

Each <application object name> is either a(n) <subtype application object name 1>, or a(n) <subtype application object name 2>, ... or a(n) <subtype application object name n>.

Each application object which has defined categorisations of the first form also results in the simple application object test purpose described in B.1.1.

This second form of categorisation definition describes an abstract supertype; as such the supertype instance is never to be created as itself. Therefore, each application object which has defined categorisations of the second form does not result in the simple application object test purpose described in B.1.1.

Each defined categorisation of either form results in one additional test purpose associated with the supertype application object, corresponding to an instance of that object created as a supertype component of the named categorisation subtype instance. This test purpose consists of the keyword “ae”, the test-purpose-id, and the phrase supertype application object name as subtype application object name.

EXAMPLE 28 - In ISO 10303-203 4.2.15 Geometric_model_representation application object has the following categorisation:

Each Geometric_model_representation may be one of the following: Advanced_B_Rep (see 4.2.2), a Faceted_B_Rep (see 4.2.14), a Non_topological_surface_and_wireframe (see 4.2.18), a Manifold_surface_with_topology (see 4.2.16), or Wireframe_with_topology (see 4.2.39).

This text results in the following test purposes

— ae001 Geometric_model_representation

— ae002 Geometric_model_representation as Advanced_B_Rep

— ae003 Geometric_model_representation as Faceted_B_Rep

— ae004 Geometric_model_representation as Non_topological_surface_and_wireframe

— ae005 Geometric_model_representation as Manifold_surface_with_topology

— ae006 Geometric_model_representation as Wireframe_with_topology

EXAMPLE 29 - In ISO 10303-203 4.2.24 Work_request application object has the following categorisation:

Each Work_request is either a Start_request (see 4.2.33) or a Change_request (see 4.2.6).

This text results in the following test purposes

— ae010 Work_request as Start_request

— ae011 Work_request as Change_request

An application object may define more than one categorisation. If an application object has at least one categorisation that defines it as an abstract supertype, then the simple application object test purpose described in B.1.1 shall not be generated, regardless of any other categorisations.

If an application object has two or more categorisations, combinations of these elements may be specified in test purposes. These test purposes represent an instance created as an AND combination of two different categorisation subtypes. The specification of such complex test purposes is an option available to the developer of the abstract test suite, but is not required.

EXAMPLE 30 - In ISO 10303-201 4.2.28 Draughting_annotation application object has two separate categorisations. Each Draughting_annotation is either a Annotation_element, or a Dimension, or a Draughting_callout. Each Draughting_annotation may be one of the following: Model_placed_annotation, or a Sheet_placed_annotation, or a View_placed_annotation. This results in the following mandatory test purposes:

— ae020 Draughting_annotation as Annotation_element

— ae021 Draughting_annotation as Dimension

— ae022 Draughting_annotation as Draughting_callout

— ae023 Draughting_annotation as Model_placed_annotation

— ae024 Draughting_annotation as Sheet_placed_annotation

— ae025 Draughting_annotation as View_placed_annotation

This may also result in the following optional test purposes:

— ae026 Draughting_annotation as (Annotation_element AND Model_placed_annotation)

— ae027 Draughting_annotation as (Annotation_element AND Sheet_placed_annotation)

— ae028 Draughting_annotation as (Annotation_element AND View_placed_annotation)

— ae029 Draughting_annotation as (Dimension AND Model_placed_annotation)

— ae030 Draughting_annotation as (Dimension AND Sheet_placed_annotation)

— ae031 Draughting_annotation as (Dimension AND View_placed_annotation)

— ae032 Draughting_annotation as (Draughting_callout AND Model_placed_annotation)

— ae033 Draughting_annotation as (Draughting_callout AND Sheet_placed_annotation)

— ae034 Draughting_annotation as (Draughting_callout AND View_placed_annotation)

Where such combinations are not documented as explicit test purposes, the developers of an abstract test suite shall nonetheless indicate where such combinations exist. Each of the categorisation subtypes for an application object must appear in at least one of the mandatory test purposes even if it is a combination of more than one categorisation subtype. In example 30, the subtype, model_placed_annotation only appears in combination with another subtype. The test case developer must include one of the optional combination test purposes, ae026, ae029, or ae032 in the list of AE test purposes.

In the subclauses of 4.2 of the AP, an application object may define implicit categorisations using the following:

The types of <application object name> that may be specified are: <type 1>, <type 2>, ... <type n>.

Such implicit categorisations are found where an application object is defined as having several values or types, where these are not themselves stated explicitly as separate application objects or as attribute values. These may result in test purposes that are equivalent to those for explicit categorisations. Like the complex combination test purposes, the specification of such test purposes is an option available to the developer of the abstract test suite, but is not required.

EXAMPLE 31 - ISO 10303-201 includes an application object 2D_geometric_element, which is defined (in normative text) to be a cartesian_point, a point_on_curve, a line, a polyline, etc. ISO 10303-301 includes the test purposes:

— ae023 2D_geometric_element as cartesian_point

— ae024 2D_geometric_element as point_on_curve

— ae025 2D_geometric_element as straight_line

— ae026 2D_geometric_element as composite_curve

— ae027 2D_geometric_element as bspline_curve

— ae028 2D_geometric_element as offset_curve

— ae029 2D_geometric_element as polyline

— ae001 2D_geometric_element as circle

— ae030 2D_geometric_element as ellipse

— ae031 2D_geometric_element as trimmed_conic

NOTE 5 - Application elements are documented with the same convention used in the application protocol. The first letter of the first word of an application object name is capitalized. The implicit categorisations are documented with lower case to distinguish them from application objects.

In the subclauses of 4.2 of the AP, application objects that are a subtype of another application object have a first sentence with the following format:

A(n) <application object name> is a type of <supertype application object name> that is <rest of definition>.

No additional test purpose is derived from this sentence in the AP, since an appropriate test purpose is derived from the categorisation definition provided for the supertype application object.

B.2 Test purposes derived from application object attributes

All attributes associated with an application object result in additional test purposes associated with the application object test purpose. There will be at least one test purpose for each attribute of each application object in the AP. Each attribute of an application object is listed in the application object definition in the subclause of 4.2 of the AP in a sentence of the form:

The data associated with a(n) <application object name> are the following:

— <attribute name>.

Each attribute is defined in a subclause under an application object definition except in the case where there is only one attribute. When there is only one attribute, the attribute definition follows the above sentence in the application object definition subclause. Attribute test purposes are generated according to the following specification.

B.2.1 Test purposes derived from mandatory simple type attributes

Attribute definitions are provided in the form:

The <attribute name> specifies <definition>.

A mandatory simple attribute has no additional definitions. Every mandatory attribute of an application object defined in the application protocol results in one additional test purpose for that attribute.

EXAMPLE 32 - In ISO 10303-201, 4.2.30.2 Drawing_number

The Drawing_number specifies the identification of a particular drawing by an organization.

results in the following additional test purpose:

— ae032 Drawing with Drawing_number.

B.2.2 Aggregate type attributes

Aggregated attribute definitions are indicated by an additional sentence of the form:

There may be more than one <attribute name> for a <application object name>.

Every aggregate attribute of an application object defined in the application protocol results in at least one additional test purpose for that attribute.

An aggregation results in one test purpose for the attribute with one member of the aggregation, and one test purpose for the upper limit of the aggregation.

EXAMPLE 33 - In ISO 10303-201, the aggregated attribute Contract_reference with one or many members. ISO 10303-301 results in the test purposes:

— ae036 Drawing with Contract_reference of one element.

— ae037 Drawing with Contract_reference of many elements.

NOTE 6 - In ISO 10303-201, the aggregated attribute Contract_reference is also defined as an optional attribute (see B.2.4). Therefore another test purpose results:

— ae038 Drawing with Contract_reference not present.

An aggregation with many members, defined as having a lower bound greater than one, results in a single test purpose.

If an explicit upper limit is specified for the aggregation then the AE test purpose still uses the words, "of many elements" as shown in example 33. If necessary a domain test purpose may specify a specific number of elements. An aggregation with a fixed number of members results in a single test purpose with the words "of <n> elements" where <n> is replaced by the fixed number of members.

B.2.3 Specific attribute values

Every application object that has one or more required values for an attribute results in one or more additional test purposes. One test purpose is specified for each discrete value of the attribute identified in the application protocol.

EXAMPLE 34 - In ISO 10303-203, 4.2.20 Part_version has the attribute 4.2.20.3 Release_status is defined to always be one of two values: released or unreleased. This results in the following test purposes:

— ae040 Part_version with release_status = released.

— ae041 Part_version with release_status = unreleased.

An attributes of an application object that has an explicit map to an AIM entity attributes of type BOOLEAN, LOGICAL, and ENUMERATION results in one test purpose for each possible value.

EXAMPLE 35 - An application protocol 2[??] includes an application object B with an attribute X, which maps to an AIM attribute, whose value is BOOLEAN. The abstract test suite contains the following test purposes:

— ae042/atc094 B with X = TRUE.

— ae043/atc094 B with X = FALSE.

NOTE 7 - Attributes that reference a closed range of integer, character or string values may result in one test purpose for each possible value. The AP/ATS development team shall determine the cases where an attribute has a set of required values rather than a range within a bounded or unbounded type. It is not intended, for example, that an attribute defined as having a value between 0 and 1000 have a test purpose for each intermediate value! It is suggested that such test purposes arise only where each value of the attribute gives a different *meaning* to the application object.

B.2.4 Optional attributes

Optional attributes are indicated by a sentence of the form:

The <attribute name> need not be specified for a particular <application object name>.

Every optional attribute of an application object defined in the application protocol results in one additional test purpose for that attribute, corresponding to the case of option not present. Each attribute type described in the sections above may also be an optional attribute.

EXAMPLE 36 - In ISO 10303-203, 4.2.25 Planned_sequence_effectivity has an optional attribute 4.2.25.4 thru_effectivity_id. This results in the test purposes:

— ae044 Planned_sequence_effectivity with thru_effectivity_id.

— ae045 Planned_sequence_effectivity with thru_effectivity_id not present.

Aggregate attributes will generate test purposes corresponding to the bounds of the aggregate as described in section B.2.2 above (see example 33).

Attributes having specific values result in a test purpose corresponding to each discrete value of the aggregate as described in section B.2.3 above.

EXAMPLE 37 - An application protocol 2[??] includes an application object B with an optional attribute X, which maps to an AIM attribute of type BOOLEAN. The abstract test suite contains the following test purposes:

— ae046 B with X = TRUE.

— ae047 B with X = FALSE.

— ae048 B with X not present.

B.3 Test purposes derived from application assertions

Each relationship between a pair of application objects is documented once in a subclause of 4.3 of the AP. The primary relationship is listed first. Test purposes derived from the primary relationship follow the test purposes for the attributes of the primary application object. Inverse relationships are stated in a second sentence of the application assertion subclause. Test purposes derived from the inverse

relationship follow the test purposes for the attributes of the secondary application object. Assertions are written in the form:

> Each <application object name> <role description> (zero, one, or many | one or more | exactly one) <application object name> [objects].

The relationship is described with a cardinality of zero, one, and/or many.

— Each application assertion having cardinality zero results in a single test purpose stating the assertion name modified with the words "<role description> zero".

— Each application assertion having cardinality one results in a single test purpose stating the assertion name modified with the phrase "<role description> one".

— Each application assertion having cardinality many results in a single test purpose stating the assertion name modified with the phrase "<role description> many".

EXAMPLE 38 - In ISO 10303-203, 4.3.11 Part to Alternate_part is defined as

Each Part has an alternate of zero, one, or many Alternate_part objects. Each Alternate_part is an alternate for one or many Part objects.

This results in the test purposes for the application object Part:

— ae049 Part has an alternate of zero Alternate_part.

— ae050 Part has an alternate of one Alternate_part.

— ae051 Part has an alternate of many Alternate_part objects.

and the test purposes for the application object Alternate_part

— ae052 Alternate_part is an alternate for one Part.

— ae053 Alternate_part is an alternate for many Part objects.

There may be more than one relationship between a pair of application objects. In this case, the subclause of 4.3 contains two pairs of assertions.

EXAMPLE 39 - In ISO 10303-201, 4.3.26 Dimension to Dimension_sequence_pair is defined as

Each Dimension is the predecessor for zero, one, or many Dimension_sequence_pair objects. Each Dimension_sequence_pair has as a predecessor exactly one Dimension. Each Dimension is the successor of zero, one, or many Dimension_sequence_pair objects. Each Dimension_sequence_pair has as a successor exactly one Dimension.

This results in the test purposes for application object Dimension:

— ae054 Dimension is the predecessor for zero Dimension_sequence_pair.

— ae055 Dimension is the predecessor for one Dimension_sequence_pair.

— ae056 Dimension is the predecessor for many Dimension_sequence_pair objects.

— ae057 Dimension is the successor for zero Dimension_sequence_pair.

— ae058 Dimension is the successor for one Dimension_sequence_pair.

— ae059 Dimension is the successor for many Dimension_sequence_pair objects.

And the test purposes for the application object Dimension_sequence_pair:

— ae060 Dimension_sequence_pair has as a predecessor one Dimension.

— ae061 Dimension_sequence_pair has as a successor one Dimension.

Annex C
(normative)

AIM test purpose development

Application interpreted model (AIM) test purposes are derived from the normative AIM EXPRESS schema defined in an application protocol. This annex describes a process to generate an explicit, comprehensive list of AE test purposes from the EXPRESS schema contained in annex A of the associated AP.

The interpretation of each test purpose derived from the AIM EXPRESS schema is given in the following statement:

> Correctly instantiate in the implementation under test the AIM elements associated with the unique AIM entity corresponding to *(insert test purpose here)* in at least one test case within the test suite.

This annex provides the details of how to generate the test purposes from the EXPRESS schema of an AP's application interpreted model (AIM).

C.1 Test purposes derived from entities

Every entity is associated with at least one explicit AIM test purpose. This entity test purpose implies all required features of the entity are present and correct in the instantiation. This includes all mandatory simple type attributes, all mandatory entity type attributes, all derive and inverse attributes, all local constraints (UNIQUE and WHERE), and all applicable rules.

C.1.1 Simple entities

Each entity defined in the AIM EXPRESS results in one explicit test purpose. This test purpose corresponds to an instance of that entity as defined itself, not as a defined subtype (see C.2.2).

EXAMPLE 40 — The entity:

```
ENTITY simple_entity
  attr_1 : INTEGER;
  attr_2 : other_entity;
UNIQUE
  UR1 : attr_1;
WHERE
  WR1 : where_expression;
END_ENTITY;
```

has the test purpose:

— aim001 simple_entity.

C.1.2 Entities with subtypes

Subtypes of an entity defined using the ONEOF relationship, and subtype instances resulting from a ONEOF instantiation of an ANDOR relationship, do not require a test purpose. The relevant test purposes for these subtypes are generated from the EXPRESS entity definitions of the corresponding subtypes within the AIM schema of interest.

Each entity having more than one subtype defined using the AND or the ANDOR relationship has subtypes that may be combined. If an entity is defined as having subtypes that may be combined, then the complex instantiation of specific subtype combinations may result in test purposes. The specification of such test purposes is an option available to the developer of the abstract test suite, but is not required.

EXAMPLE 41 — The entity:

```
ENTITY entity_with_subtypes
  SUPERTYPE OF (subtype_1, subtype_2);
  attr_1 : INTEGER;
END_ENTITY;
```

may have the specific test purpose:

— aim002 entity_with_subtypes as (subtype_1 AND subtype_2).

C.2 Test purposes derived from attributes of entities

All attributes types referenced in this clause result in additional test purposes associated with the entity test purpose. Attribute test purposes are associated with every entity test purpose derived from an entity having attributes. These test purposes are generated according to the following specification.

C.2.1 Entities with aggregation type attributes

Each Aggregate type attribute in an entity results in the definition of at least one explicit test purpose associated with that attribute.

An aggregation with zero or one members (defined with bounds [0:1]) results in one test purpose for the attribute with zero members of the aggregation, and one test purpose for the attribute with one member of the aggregation, i.e. the upper limit.

EXAMPLE 42 — The entity:

```
ENTITY entity_with_aggr;
  array_attr : ARRAY [0:1] of base_type;
END_ENTITY;
```

has the following test purposes:

— aim003 entity_with_aggr with array_attr of zero elements;

— aim004 entity_with_aggr with array_attr of one element.

An aggregation of zero, one or many elements results in one test purpose for the attribute with zero elements of the aggregation, one test purpose for the attribute with one element of the aggregation, and one test purpose for the upper limit of the aggregation.

EXAMPLE 43 — The entity:

```
ENTITY entity_with_aggr;
  set_attr : SET [0:?] of base_type;
END_ENTITY;
```

has the following test purposes:

— aim005 entity_with_aggr with set_attr of zero elements;

— aim006 entity_with_aggr with set_attr of one element;

— aim007 entity_with_aggr with set_attr of many elements.

An aggregation with one or many members results in one test purpose for the attribute with one member of the aggregation, and one test purpose for the upper limit of the aggregation.

EXAMPLE 44 — The entity:

```
ENTITY entity_with_aggr;
  list_attr : LIST [1:10] of base_type;
END_ENTITY;
```

has the following test purposes:

— aim008 entity_with_aggr with list_attr of one element;

— aim009 entity_with_aggr with list_attr of many elements.

An aggregation with many members results in one test purpose for the upper limit of the aggregation.

EXAMPLE 45 — The entity:

```
ENTITY entity_with_aggr;
  bag_attr : BAG [2:?] of base_type;
END_ENTITY;
```

has the following test purpose:

— aim010 entity_with_aggr with bag_attr of many elements.

C.2.2 Entities with BOOLEAN type attributes

Each attribute of type BOOLEAN in an entity results in the definition of two additional test purposes, associated with TRUE and FALSE values assigned to the attribute.

EXAMPLE 46 — The entity:

```
ENTITY entity_with_boolean;
  boolean_attr : BOOLEAN;
  attr_2 : INTEGER;
```

```
  attr_3 : REAL;
END_ENTITY;
```

has two test purposes derived from the boolean attribute boolean_attr:

— aim011 entity_with_boolean with boolean_attr = TRUE;

— aim012 entity_with_boolean with boolean_attr = FALSE.

C.2.3 Entities with LOGICAL type attributes

Each attribute of type LOGICAL in an entity results in the definition of three additional test purposes, associated with TRUE, FALSE and UNKNOWN values assigned to the attribute.

EXAMPLE 47 — The entity:

```
ENTITY entity_with_logical ;
  logical_attr : LOGICAL;
  attr_2 : INTEGER;
  attr_3 : REAL;
END_ENTITY;
```

has three test purposes derived from the logical attribute logical_attr:

— aim013 entity_with_logical with logical_attr = TRUE;

— aim014 entity_with_logical with logical_attr = FALSE;

— aim015 entity_with_logical with logical_attr = UNKNOWN.

C.2.4 Entities with ENUMERATION type attributes

Each attribute of an ENUMERATION type in an entity results in the definition of one additional test purpose for each value in the enumeration list. Typically, the number of possible specific attribute values is finite and small. However, there may be a large number of enumerated values. The AP/ATS development team will have to determine the cases where an enumerated value carries a specific meaning in the context of the application protocol.

EXAMPLE 48 — Given:

```
TYPE e_type = ENUMERATION OF (e1, e2, e3, e4)
END_TYPE;
```

the entity:

```
ENTITY entity_with_enumeration ;
  enum_attr : e_type;
  attr_2 : INTEGER;
  attr_3 : REAL;
END_ENTITY;
```

has four test purposes derived from the enumeration attribute enum_attr:

— aim016 entity_with_enumeration with enum_attr = e1;

— aim017 entity_with_enumeration with enum_attr = e2;

— aim018 entity_with_enumeration with enum_attr = e3;

— aim019 entity_with_enumeration with enum_attr = e4.

C.2.5 Entities with SELECT type attributes

Each attribute of a SELECT type in an entity results in the definition of one additional test purpose for each type specified in the select list. Since the number of possible specific attribute value types is finite and typically small, a heuristic of testing all possible value types has been adopted.

EXAMPLE 49 — Given:

```
TYPE s_type = SELECT (s1, s2)
END_TYPE;

TYPE s2_type = SELECT (s3, s4)
END_TYPE;
```

the entity:

```
ENTITY entity_with_select ;
  select_attr : s_type;
  attr_2 : INTEGER;
  attr_3 : REAL;
END_ENTITY;
```

has three test purposes derived from the select attribute select_attr:

— aim020 entity_with_select with select_attr as s1;

— aim021 entity_with_select with select_attr as s3;

— aim022 entity_with_select with select_attr as s4.

C.2.6 Entities with OPTIONAL attributes

Each OPTIONAL attribute in an entity results in the definition of at least two additional test purposes associated with the presence or absence of the attribute.

In all cases, an optional attribute contains one test purpose corresponding to the option not present. At least one other test purpose will correspond to the case of option present. Note that each attribute type described in the sections above may also be an optional attribute. As described above, depending upon the type of the attribute, more than one test purpose may be required for the option present case. Attributes that reference a simple type will contain a single test purpose representing the case of present.

EXAMPLE 50 - The entity:

```
ENTITY entity_with_optional ;
  opt_attr : OPTIONAL INTEGER;
  attr_2 : INTEGER;
  attr_3 : REAL;
END_ENTITY;
```

has two test purposes derived from the optional attribute opt_attr:

— aim023 entity_with_optional with opt_attr not present;

— aim024 entity_with_optional with opt_attr.

Aggregate attributes will contain result in a test purpose corresponding to the bounds of the aggregate as described in section C.2.2 above.

EXAMPLE 51 - The entity:

```
ENTITY entity_with_optional ;
  opt_attr : OPTIONAL SET [0:?] of base_type;
  attr_2 : INTEGER;
  attr_3 : REAL;
END_ENTITY;
```

has four test purposes derived from the optional attribute opt_attr:

— aim025 entity_with_optional with opt_attr not present;

— aim026 entity_with_optional with opt_attr of zero elements;

— aim027 entity_with_optional with opt_attr of one element;

— aim028 entity_with_optional with opt_attr of many elements.

Attributes having specific values will contain one present test purpose corresponding to each discrete value of the aggregate as described in C.2.3, C.2.4, and C.2.

EXAMPLE 52 — The entity:

```
ENTITY entity_with_opt_boolean ;
  boolean_attr : OPTIONAL BOOLEAN;
  attr_2 : INTEGER;
  attr_3 : REAL;
END_ENTITY;
```

has three test purposes derived from the optional attribute boolean_attr:

— aim029 entity_with_opt_boolean with boolean_attr not present;

— aim030 entity_with_opt_boolean with boolean_attr = TRUE;

— aim031 entity_with_opt_boolean with boolean_attr = FALSE.

C.3 CONSTANTS, TYPES, FUNCTIONS, PROCEDURES, and RULES

EXPRESS CONSTANTS, TYPES, FUNCTIONS, PROCEDURES, and RULES have of themselves no corresponding test purposes.

C.3.1 Removal of AIM test purposes

EXPRESS RULES (global constraints), FUNCTIONS, and PROCEDURES that act as local constraints, and INVERSE attributes may disallow certain AIM structure. This may disallow the instantiation of entities of a particular type, affect the cardinality of an aggregate attribute, restrict the range of values that an ENUMERATION, LOGICAL, or BOOLEAN type attributes may assume, or constrain the instantiated model in some other way. Any test purpose generated by the process described in C.2 and C.3 below that indicates an AIM structure in violation of any defined EXPRESS constraint shall not be included in the resultant set of AIM derived test purposes.

EXAMPLE 53 - In ISO 10303-203, 5.2.5.74 subtype_mandatory_representation specifies a EXPRESS RULE which requires that each instance of representation will be a shape_representation. This global constraint will caues the test purpose

aim[???] representation

not to be included in the final list of test purposes for ISO 10303-203.

Annex D
(normative)

Derivable verdict criteria development

This annex provides the algorithms for deriving preprocessor and postprocessor verdict criteria.

D.1 Deriving preprocessor verdict criteria from the AP mapping table

The preprocessor input specification basically contains three different kinds of entries:

— Entries that specify that an application object's instance is presented to the preprocessor as input;

— Entries that specify values for an application object's attribute;

— Entries that specify links between two application object instances (assertions).

To derive the AIM verdict criteria for these kinds of entries, first the corresponding mapping table entries need to be identified. Clause 5 of the AP presents the mapping tables grouped by Unit of Functionality, and the proper mapping tables are therefore dependent on the conformance class an ATC is related to and possibly additional information which has to be present in the description of the ATC. For each entry in the preprocessor input specification, the corresponding entry in the AP's mapping table is identified first. The verdict criteria are then derived from this mapping table entry.

D.1.1 Verdict criteria corresponding to application objects

The first kind of entries corresponds to the capitalized entries in the mapping table. In column 2 of the mapping table, each of these entries gives an AIM element which we call the "corresponding AIM instance" of the application element. The set of all preprocessor input specification entries of this kind therefore selects a set of corresponding AIM instances within the set of all AIM instances in the expected output of the preprocessor. Whenever a verdict criterion is related to such an AIM instance, this is expressed by adding the phrase "that corresponds to @<n>" to the phrase referencing the AIM instance, where @<n> is the preprocessor input specification entry of the application element.

Column 2 of the mapping table describes the mapping of an application object by stating the corresponding AIM element. This results in the following verdict criterion:

—An instance of <AIM element> that corresponds to @<n> is present.

Column 2 may also contain a set of alternatives in the form

```
(aim element 1)
(aim element 2)
...
(aim element n)
```

These alternatives are expressed in the following verdict criterion:

— An instance of <aim element 1> or <aim element 2> ... or <aim element n> that corresponds to @<n> are present.

Note 8 - The phrase "instance of x" in the context of these verdict criteria is to be interpreted as "instance of x or one of its subtypes".

Column 5 of the mapping table may contain a reference path for the application object. This results in additional derivable verdict criteria which are presented after the verdict criterion given above in an indented style. The details of deriving these verdict criteria are given in D.1.3.

In the preprocessor input specification table, additional entries may follow the entry that relates to the application object instance which specify attribute values or links to other application objects. The details how to construct verdict criteria for these are given in the following subsections. All these additional verdict criteria are to be indented relative to the first verdict criterion related to the application object as given above, in order to denote that they all relate to the corresponding AIM instance identified when evaluating the first verdict criterion.

D.1.2 Verdict criteria for attributes of application objects

In the case of an application object's attribute, column 2 of the mapping table specifies the attribute or AIM entity where the application object's attribute value is to be found, and column 5 specifies an optional reference path that establishes a link from the corresponding AIM instance of the application object to the item given in column 2. Therefore, if a reference path is given in column 5, the verdict criteria derivable from it are to be presented first according to the method described in D.1.3. One additional verdict criterion is then presented in both cases to ensure that the application object's attribute value was encoded properly.

If column 2 contains an entry of the form "`aim_element.attribute`", the basic phrase for the verdict criterion will be:

—the attribute `attribute` of the instance of `aim_element` [that corresponds to @<n>] has the value @<v>.

Note 9 - The phrase within brackets is only required if the AIM instance belongs to the set of corresponding AIM instances described in D.2. The phrase compares the given attribute value to the value specified in either the suggested values or the mandatory values column of the preprocessor input specification. During the creation of an executable test case, the testing laboratory will assign a fixed value for the table entry @<v> which then is promoted to the verdict criterion.

If column 2 contains an entry of the form "`aim_element`", the basic phrase will be:

—the instance of `aim_element` [that corresponds to @<n>] encodes the value @<v>.

The actual verdict criterion is constructed from the above phrase as follows:

— if column 2 of the mapping table contains a single entry, it is: " *phrase*";

— if column 2 of the mapping table contains several alternatives in the form

```
(alternative 1)

(alternative 2)

...

(alternative n)
```

thereby indicating that an attribute can be mapped in different ways, the verdict criterion is: "*phrase for alternative1*, or *phrase for alternative 2*, ... or *phrase for alternative n*".

D.1.3 Verdict criteria corresponding to application assertions

In the case of an application assertion, two base cases are to be distinguished:

— Identical mapping: Column 2 of the mapping table contains the keyword `IDENTICAL MAPPING`, which means that the two application objects involved in the assertion are represented by a single AIM entity. In this case no extra verdict criteria are required besides those that were created for the two application objects according to D.1.1;

— Path mapping: Column 2 of the mapping table contains the keyword `PATH`. This means that Column 5 of the mapping table describes the path through the instance graph from the AIM instance corresponding to the relating application object to the AIM instance corresponding to the related application object. A series of verdict criteria is derived to ensure that the whole path was constructed properly by the preprocessor. In the cases described in D.1.1 and D.1.2 where column 5 was not empty, verdict criteria are derived in the same fashion.

The following describes an algorithm for AIM verdict criteria generation from mapping paths. A path in the mapping table may contain three kinds of operators which imply forks in the path:

— OR (denoted by the syntax "`(alternative 1)(alternative 2)`");

— AND (denoted by the syntax "`[requirement 1][requirement 2]`");

— mapping rules (denoted by the syntax "`{rule-requirement}`").

In order to make the derivation of verdict criteria easier, first eliminate the OR-operators:

Algorithm step 1: OR elimination.

Generate, for each OR alternative, a copy of the path where one of the branches replaces the OR-construct. If the mapping path contains several points where an OR operator introduces a fork in the path, do the following:

— if the mapping table makes use of the "#n"-syntax to clarify which branches belong together, use this information for generation of the path copies;

— if it does not, repeat the OR-elimination with the copies until all OR-operators are deleted. This has the consequence that all possible combinations of the branches are generated.

Subsequent steps of the algorithm are applied to the path copies separately. The basic verdict criterion has the following general format:

Does the following set of conditions apply and are satisfied?

verdict criteria derived from copy 1

Or does the following set of conditions apply and are satisfied?

verdict criteria derived from copy 2 …

Or is it the case that the following set of conditions is satisfied?

verdict criteria derived from copy n.

The verdict criteria derived from the copies are indented relatively to the basic verdict criterion (indent level 2) to clarify their relation to it.

The mapping path describes the relations between entities in a very explicit manner. SELECT trees are described in a verbose manner, and instead of using the inheritance mechanisms of EXPRESS, supertype/subtype relations have to be stated explicitly (i.e., the mapping table cannot directly relate to an inherited attribute *attr* of an entity *entity*, but has to state the equivalent of "entity is a subtype of super_entity, and super_entity's attribute attr is to be used such and such").

In order to avoid the generation of redundant verdict criteria, the following preprocessing steps are applied to the path:

Algorithm step 2: Elimination of select paths.

Delete all mapping table lines of the form `"select_type = selected_item"`. After having deleted a consecutive set of lines, analyze the remaining lines above and below as follows:

— if the line above contains a SELECT type, propagate the entity or type from the line below to this line;

— if the line below contains a SELECT type, propagate the entity or type from the line above to this line.

Preprocessing step 3: Elimination of supertype/subtype statements.

Identify lines in the mapping table which contain "<=" (is subtype of) or "=>" (is supertype of) statements. For each occurrence, identify the set of entities that is linked by a chain of "<=" or "=>" statements (mapping rules do not interrupt a chain, but can add entities to the set. At AND operators, a chain can have forks to or from the branches of the AND operator). From this set, iteratively delete the entities which are supertypes of all remaining other entities.

Note that usually a single entity will remain in the set which is required to be instantiated. If more than one entity remains, this means that a complex instance is required to be instantiated.

Replace all occurrences of entities from the original set within the chain and mapping rules included in the chain by the remaining entity set. Apply this also to the lines above the chain's start line (or start lines, in the case of AND-operators) and the lines below the chain's end line(s), as long as one of the entities from the original set is present in these neighbouring lines.

Eliminate all lines with "<=" or "=>" operators, and eliminate those mapping rules in the chain which do not contain references to attributes.

For AND operators, delete all lines from the branches which became identical for all branches during the process. If the remaining lines of the branches are identical for all (or if no lines remain), chose one branch and delete the whole AND construct. If they differ, eliminate the AND operators.

After these steps, the verdict criteria can be derived according to the following table:

Mapping table entry	Resulting verdict criterion
`entity`	An instance of *entity* is present.
`Entity.attrib -> other_entity`	The attribute *attrib* of the instance of *entity* references an instance of *other_entity*.
`Entity.attrib[i] -> other_entity`	The attribute *attrib* of the instance of *entity* references an instance of *other_entity*.
`Entity.attrib[n] ->other_entity`	The attribute *attrib*'s *n*-th component of the instance of *entity* references an instance of *other_entity*.
`Entity <- other_entity.attrib`	The instance of *entity* is referenced by an instance of *other_entity* through attribute *attrib*.
`Entity <- other_entity.attrib[i]`	The instance of *entity* is referenced by an instance of *other_entity* through attribute *attrib*.
`Entity <- other_entity.attrib[n]`	The instance of *entity* is referenced by an instance of *other_entity* through attribute *attrib*'s *n*-th component.
`Entity.attrib = val`	The attribute *attrib* of the instance of *entity* has the value *val*.
`Entity.attrib[i] = val`	The attribute *attrib* of the instance of *entity* has the value *val* for one of its components.
`Entity.attrib[n] = val`	The attribute *attrib* of the instance of *entity* has the value *val* for its *n*-th component.

NOTES

4 - phrases of the form "instance of entity" are always to be enhanced by "that corresponds to @<n>" where appropriate

5 - Where the supertype/subtype elimination process replaced an entity by a set of more than one entity, the phrase "instance of entity" is to be replaced by "complex instance containing instances of *set-element 1*, ..., and *set-element n*"

6 - Apply this table line by line. If a mapping rules is encountered, and the line before the mapping rules ends in "->" or "<-", first generate the verdict criterion for the pair of lines before and after the mapping rules, then for the lines within the mapping rule.

7 - If a verdict criterion is constructed from two lines which are related by "<-" or "->" operators, do not re-use either of them for the derivation of another verdict criterion unless both lines contain "->" or "<-" operators.

D.2 Deriving postprocessor verdict criteria from the expected output specification

The expected output specification is the realisation of the preprocessor input specification for a given test case. The expected output specification (EOS) basically contains three different kinds of entries:

— entries that specify that an application object's instance is expected;

— entries that give specific values for an application object's attribute which are expected;

— entries that specify that links between two application object instances (assertions) are expected.

Verdict criteria are derived from the EOS for those entries that are marked by an asterisk in the V column in order to denote that they correspond to the AIM test purposes identified for the abstract test case.

The following table lists the constructs encountered in the EOS and gives the corresponding verdict criteria.

Case	Syntax	Value	Verdict criterion
Object	@n application_element		A construct is present that corresponds to @n.
Attribute	@(n) ae.attrib	val	The equivalent of @n carries the *attrib* as *val*?
Attribute	@(n) ae.attrib[n]	val	Does the equivalent of @n carry the *attrib* with one component as *val*?
Attribute	@(n) ae.attrib	@m	Does the equivalent of @n have a construct corresponding to @m in the role of *attrib*?
Attribute	@(n) ae.attrib[n]	@m	Does the equivalent of @n have a construct corresponding to @m in the role of one *attrib*?
Assertion	@(n) ae to ae	@m	Is the equivalent of @n related to a construct corresponding to @m according to the reference path?

Annex E
(normative)

Abstract test suite document template

This annex provides some required and recommended text for use in various clauses and subclauses of abstract test suite parts of ISO 10303. The abstract test suite editor is to remove or replace, as appropriate for the particular abstract test suite, all text sections surrounded by angle brackets such as <this text is an example of what shall be removed or replaced>

E.1 Cover page

The abstract test suite shall include the cover page required of all ISO TC184/SC4 documents. The "Comments to reader" block on the cover page shall include a reference to the specific version of the AP based upon which the abstract test suite was developed.

E.2 Table of Contents

The Table of Contents includes all the first and second level subclauses in the document. For Clause 6 only, it also includes the third level subclause headings. The Table of Contents includes all the figures, tables, and annexes. It is formatted according to the Supplementary Directives.

E.3 Foreword

The abstract test suite forward text shall be taken directly from the ISO TC184/SC4 Editing Committee Supplementary Directives.

E.4 Introduction

The text in this subclause shall be used for the abstract test suite part introduction, without the indentation shown here and with the appropriate substitutions.

> ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.
>
> This International Standard is organized as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series: description methods, integrated resources, application protocols, abstract test suites, implementation methods, and conformance testing. The series are described in ISO 10303-1. This part of ISO 10303 is a member of the abstract test suite series.
>
> The purpose of an abstract test suite is to provide a basis for evaluating whether a particular implementation of an application protocol actually conforms to the requirements of that

application protocol. A standard abstract test suite helps ensure that evaluations of conformance are conducted in a consistent manner by different test laboratories.

This part of ISO 10303 specifies the abstract test suite for ISO 10303-2<??>, application protocol <?title?>. The abstract test cases presented here are the basis for conformance testing of implementations of ISO 10303-2<??>.

This abstract test suite is made up of two major parts:

— the test purposes, the specific items to be covered by conformance testing;

— the set of abstract test cases that meet those test purposes.

The test purposes are statements of the application protocol requirements that are to be addressed by the abstract test cases. Test purposes are derived primarily from the application protocol's information requirements and AIM, as well as from other sources such as standards referenced by the application protocol and other requirements stated in the application protocol conformance requirements clause.

The abstract test cases address the test purposes by:

— specifying the requirements for input data to be used when testing an implementation of the application protocol;

— specifying the verdict criteria to be used when evaluating whether the implementation successfully converted the input data to a different form.

The abstract test cases set the requirements for the executable test cases that are required to actually conduct a conformance test. Executable test cases contain the scripts, detailed values, and other explicit information required to conduct a conformance test on a specific implementation of the application protocol.

At the time of publication of this document, conformance testing requirements had been established for implementations of application protocols in combination with ISO 10303-21 and ISO 10303-22. This part of ISO 10303 only specifies test purposes and abstract test cases for a subset of such implementations.

For ISO 10303-21, two kinds of implementations, preprocessors and postprocessrs, must be tested. Both thest are addressed in this abstract test suite.

For ISO 10303-22, a class of applications will possess the capability to upload and download AP-compliant SDAI-models or schema instances to and from applications that implement the SDAI. By providing test case data that correspond with SDAI-models, this abstract test suite addresses such applications in a single-schema scenario.

E.5 Title

The title of an abstract test suite shall be:

Industrial automation systems and integration - Product data representation and exchange - Part 3<??>: Abstract test suite: <Same title as corresponding AP>

E.6 Clause 1 Scope

The text in this subclause shall be used for the abstract test suite part scope clause, without the indentation shown here and with the appropriate substitutions.

This part of ISO 10303 specifies the abstract test suite to be used in the conformance testing of implementations of ISO 10303-2<??>. The following are within the scope of this part of ISO 10303:

— the specification of the test purposes associated with ISO 10303-2<??>;

— the verdict criteria to be applied during conformance testing of an implementation of ISO 10303-2<??> using ISO 10303-21 or ISO 10303-22;

NOTE - The verdict criteria are used to ascertain whether a test purpose has been satisfactorily met by an implementation under test (IUT) within the context of a given test case.

— the abstract test cases to be used as the basis for the executable test cases for conformance testing.

The following are outside the scope of this part of ISO 10303:

— the creation of executable test cases;

— test specifications for tests other than conformance testing such as interoperability or acceptance testing;

— other implementation methods.

E.7 Clause 2 Normative references

The text in this subclause shall be used for the abstract test suite part references clause, without the indentation shown here and with the appropriate substitutions. If any other standards are specifically referenced within the abstract test suite (e.g., AICs), then they need to be represented in the references list. References that are already covered by the AP do not need to be referenced here unless test purposes are included that result directly from the other standard. For example, references to the 40 series parts shall not be necessary here, because no test purposes will result directly from those parts.

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO 10303 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 10303-1:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.*

ISO 10303-11:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 11: description methods: The EXPRESS language reference manual.*

ISO/TR 10303-12:1996, *Industrial automation systems and integration — Product data representation and exchange — Part 12: description methods: The EXPRESS-I language reference manual.* [Note: A reference to ISO/TR 10303-12 is necessary only if EXPRESS-I is used in this part.]

ISO 10303-21:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation method: Clear text encoding of the exchange structure.*

ISO 10303-22:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation method: Standard data access interface.*

ISO 10303-31:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 31: Conformance testing methodology and framework: General concepts.*

ISO 10303-32 *Industrial automation systems and integration — Product data representation and exchange — Part 32: Conformance testing methodology and framework: Requirements on testing laboratories and clients* [3].

ISO 10303-203:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 2*<??>*: Application protocol:* [Note: Insert here the title of the application protocol for which this document is the abstract test suite.]

[Note: If any other standards are specifically referenced within the ABSTRACT TEST SUITE, then they need to be represented in the references list. References that are already covered by the AP do not need to be referenced here. For example, references to ISO 10303-4x parts are not necessary here.]

E.8 Clause 3 Definitions and Abbreviations

The remaining text in this subclause shall be used for the abstract test suite part definitions clause, without the indentation shown here and with the appropriate substitutions and formatting.

3.1 Terms defined in ISO 10303-1

This part of ISO 10303 makes use of the following terms defined in ISO 10303- 1.

— abstract test suite;

— application protocol (AP);

— implementation method.

---

[3] To be published.

[Note: include any other appropriate terms from part 1.]

3.2 Terms defined in ISO 10303-31

This part of ISO 10303 makes use of the following terms defined in ISO 10303- 31.

— abstract test case;

— conformance testing;

— executable test case;

— test purpose;

— verdict criterion.

[Note: include any other appropriate terms from part 31.]

3.3 Terms defined in ISO 10303-2[??]

This part of ISO 10303 makes use of the following terms defined in ISO 10303- 2<??>.

[Note: include all appropriate terms from the relevant AP.]

3.4 Other definitions

For the purposes of this part of ISO 10303, the following definitions apply.

[Note: terms defined for this part, if any, beyond those listed by reference above.]

3.5 Abbreviations

For the purposes of this part of ISO 10303, the following abbreviations apply.

[Note: List all abbreviations used in this part.]

E.9 Clause 4 Test purposes

Test purposes shall be documented using the following clause structure:

4 Test purposes

4.1 Domain test purposes

4.2 Application element test purposes

4.2.1 Application_object_1

4.2.2 Application_object_2

...

4.2.n Application_object_n

4.3 AIM test purposes

4.3.1 Aim_entity_1

4.3.2 Aim_entity_2

...

4.3.n Aim_entity_n

4.4 External reference test purposes

4.5 Implementation method test purposes

4.6 Other test purposes

[Note: Clauses 4.4 - 4.6 shall be present only if this ATS has these classes of test purposes.]

Within this structure, the test purposes shall appear in the same order as their sources in the AP. That is, the AE test purposes shall be presented in the same order as the application objects and assertions are presented in the AP. The remaining text in this subclause shall be used as noted for the test purpose subclauses of the abstract test suite part, without the first indentation shown here and with the appropriate substitutions and additions.

For clause 4:

> This clause specifies the test purposes for this part of ISO 10303. Test purposes in 4.1 and 4.2 are derived from the information requirements contained in clause 4 of ISO 10303-2<??> and the AIM EXPRESS schema in annex A of ISO 10303-2<??>, and <Note: List other sources, if any.>. Each test purpose statement identifies some specific element from the AEs or the AIM. These test purpose statements implicitly require that the identified element, as specified in the test purpose statement, will be correctly instantiated by the implementation under test.
>
> Implementation method test purposes in 4.3 are derived from ISO 10303-21. Domain test purposes in 4.4 are derived from implicit domain requirements <Note: list source(s) of domain test purposes, examples include: other international standards referenced by the application protocol for which this document is the abstract test suite.>. Implementation method and domain test purposes are individually identified by the prefix “other” in the test purpose number. These test purposes are statements of requirements that shall be met by a conforming implementation.

For subclause 4.1:

> AE test purposes are individually identified by the prefix “ae” in the test purpose number. Each test purpose derived from the information requirements shall be interpreted as given in the following statement: the IUT shall preserve the semantic associated with the unique application element from which the test purpose was derived. This implies that the semantics of the application element are preserved by the IUT between the input and output of a test, according to the reference path specified in the mapping table of the AP. AE test purposes apply to the input specifications of

both preprocessor and postprocessor test cases. AE test purposes are derived from the AP information requirements as follows:

— application objects (see 4.2 of ISO 10303-2<??>). A test purpose derived from an application object is a simple statement of the object's name. Each application object test purpose is documented in a separate subclause.

— application objects with categorisations (subtypes) (see 4.2 of ISO 10303-2<??>). Test purposes derived from application objects with categorisations are statements of the application object name as a specific subtype.

— application object attributes (see 4.2 of ISO 10303-2<??>). Test purposes derived from application object attributes are statements of the application object name with a specific attribute name.

— application assertions (see 4.3 of ISO 10303-2<??>). Test purposes derived from application assertions are statements describing the relationship between two application objects. Application assertion test purposes address the directions of relationships as well as the number (cardinality) of relationships.

Each application object test purpose is listed as a separate subclause, with its related application object attribute test purposes and assertion test purposes.

[Note: For each application object, create a subclause, 4.1.x. Place its associated simple, categorization, attribute, and assertion test purposes in that subclause.]

For subclause 4.2:

AIM test purposes are identified by the prefix "aim" in the test purpose identifier. Each test purpose derived from the AIM EXPRESS shall be interpreted as given in the following statement: the postprocessor shall accept the input in accordance with the AIM EXPRESS structure corresponding to this test purpose. This implies that the semantics of the application element represented by the AIM element are preserved by the IUT between the input and output of a test according to the reference path specified in the mapping table of the AP. This also implies no violations of any constraints (e.g., where rules or global rules) that apply to the AIM element. AIM test purposes apply to the input specifications of postprocessor test cases only. AIM test purposes are derived directly from the expanded EXPRESS listing contained in annex A of ISO 10303-2<??> as follows:

— AIM entities. A test purpose derived from an AIM entity is a simple statement of the entity name.

— AIM entity attributes. A test purpose derived from an AIM entity attribute is a statement of the AIM entity with a given attribute.

Each AIM entity test purpose is grouped with its attribute test purposes.

[Note: For each AIM entity, create a subclause, 4.2.x. Place all its associated test purposes in that subclause.]

E.10 Clause 5 General test purposes and verdict criteria

General test purposes are developed from requirements that may be implicit in the test purposes of clause 4. General verdict criteria are associated with general test purposes and apply to all abstract test cases, all preprocessor abstract test cases, or all postprocessor abstract test cases. Verdict criteria shall be presented as statements that evaluate to pass, fail, inconclusive. The remaining text in this subclause shall be used for the abstract test suite part general verdict criteria clause, without the indentation shown here and with the appropriate substitutions.

General test purposes are statements of requirements that apply to all abstract test cases, all preprocessor abstract test cases, or all postprocessor abstract test cases. General verdict criteria are the means for evaluating whether the general test purposes are met. General verdict criteria shall be evaluated as a part of every executable test case to which they apply. Each general verdict criterion includes a reference to its associated test purpose.

5.1 General test purposes

following are the general test purposes for this part of ISO 10303:

g1 The output of an IUT shall preserve all the semantics defined by the input model according to the reference paths specified in the mapping table defined in clause 5 of ISO 10303-2<??>.

g2 The output of a preprocessor shall conform to the implementation method to which the IUT claims conformance.

g3 The instances in the output of a preprocessor shall be encoded according to the AIM EXPRESS long form and mapping table as defined in Annex A and clause 5 of ISO 10303-2<??>.

g4 A postprocessor shall accept input data which is encoded according to the implementation method to which the IUT claims conformance.

g5 A postprocessor shall accept input data structured according to the AIM EXPRESS long form and the mapping table as defined in clause 5 of ISO 10303-2<??>.

[Note: The above list may be supplemented with other general test purposes deemed necessary by the ATS developers.]

5.2 General verdict criteria for all abstract test cases

The following verdict criteria apply to all abstract test cases contained in this part of ISO 10303:

gvc1 The semantics of the input model are preserved in the output of the IUT according to the reference paths specified in the mapping table defined in clause 5 of ISO 10303-2<??> (g1).

[Note: List here all general verdict criteria that apply to every abstract test case.]

5.3 General verdict criteria for preprocessor abstract test cases

The following verdict criteria apply to all preprocessor abstract test cases contained in this part of ISO 10303:

gvc2 The output of a preprocessor conforms to the implementation method to which the IUT claims conformance (g2).

gvc3 The instances in the output of a preprocessor are encoded according to the AIM EXPRESS long form and mapping table as defined in Annex A and clause 5 of ISO 10303-2[??] (g3).

[Note: The above list may be supplemented with other general preprocessor verdict criteria deemed necessary by the ATS developers.]

5.4 General verdict criteria for postprocessor abstract test cases

The following verdict criteria apply to all postprocessor abstract test cases contained in this part of ISO 10303:

gvc4 The postprocessor accepts input data which is encoded according to the implementation method to which the IUT claims conformance (g4).

gvc5 The postprocessor accepts input data which is structured according to the AIM EXPRESS long form and mapping table as defined in Annex A and clause 5 of ISO 10303-2<??> (g5).

[Note: The above may be supplemented with other general postprocessor verdict criteria deemed necessary by the ATS developers.]

E.11 Clause 6 Abstract test cases

The text in this subclause shall be used as the introduction for the abstract test cases clause, without the indentation shown here and with the appropriate substitutions. The format for the presentation of abstract test cases is defined in clause 8 and a template for individual abstract test cases is presented in annex F.

This clause specifies the abstract test cases for this part of ISO 10303. Each abstract test case addresses one or more test purposes from clause 4. All the test purposes addressed by the test case are referenced either explicitly, in the test purposes covered sections, or indirectly, through the verdicted rows of the preprocessor input specification table.

The abstract test cases are organized by <add the appropriate explanation of the abstract test case organization here>. The title for an abstract test case signifies <add the significance of or reasoning behind the individual titles.>. All abstract test case names are unique within this part of ISO 10303.

Each abstract test case has a subclause for the preprocessor test information and a subclause for each postprocessor input specification and related test information. The preprocessor and postprocessor input specifications are mirror images of each other, i.e. they represent the same semantic information. The preprocessor input model is presented in the form of a table with five columns:

— The Id column is used to reference application objects for assertions and categorisations. It uses the same identifier as the test purpose associated with the application element in that row of the table.

— The V column specifies whether, or not, the element in that row of the table is verdicted in

this test case. A blank indicates it is not verdicted. A '*' indicates that it is verdicted using a derived verdict criteria. The derived verdict criteria determine a number references a specific verdict criteria defined in the verdict criteria section that follows the preprocessor input specification table.

— The Application Elements column identifies the particular application element or categorisation instance that is being defined by the table. For assertions the role is specified in parenthesis.

— The Value column specifies a specific value for the application element. For application objects and attributes the value column defines the semantic value for that element's instance in the input model. A "#<number>" in the column is a reference to an entity instance name in the postprocessor input specification where the corresponding value is specified. For assertions, this column holds a link to the related application object. For categorisations, the Value column identifies the subtype application object. A "<not_present>" indicates that the application element or categorisation is not present in the input model.

— The Req column specifies whether the value in the Value column is mandatory (M), suggested (S) or constrained (Cn) where 'n' is an integer referencing a note that follows the table. A suggested value may be changed by a test realiser. A mandatory value may not be changed due to rules in EXPRESS, rules in the mapping table, or the requirements of the test purpose being verdicted. Each constrained value references a note labelled C<number> at the end of the preprocessor input model table and may be modified according to specific constraints specified in it.

The postprocessor input specifications are defined using ISO 10303-<??> {Note: include a reference to either ISO 10303-12 (Express-I) or ISO 10303-21, or both, as appropriate for your test suite.] The values in the postprocessor specifications are suggested unless declared mandatory or constrained by the preprocessor input table.

The abstract test case specifies all the verdict criteria which are used to assign a verdict during testing. Special verdict criteria for preprocessor and postprocessor testing are defined explicitly in each abstract test case subclause. The relevant derived verdict criteria for preprocessor and postprocessor testing are identified in the V column of the preprocessor input table.

[Note: Each abstract test case is documented as a distinct subclause. Each abstract test case also contains a unique name within the abstract test suite. Abstract test cases start with subclause 6.1 as shown below and continue sequentially. Use the abstract test case template to format your abstract test cases.]

E.12 Annex A Conformance classes

Annex A is a required normative annex that shall contain lists that show which abstract test cases are required to be executed for each conformance class of the AP. For an AP with conformance classes, a subclause of the annex shall be provided for each conformance class. Each subclause shall be of the following form, without the indentation shown here and with the appropriate substitutions of numbers:

A.1 Conformance class 1

To conform to conformance class 1 of ISO 10303-2<??>, an implementation shall pass executable versions of the following abstract test cases:

— ATC001

— ATC002

— ATC004

[etc.]

If the AP does not have any defined conformance classes, the content of this annex reduces to the following statement, without the indentation shown here:

Conformance to ISO 10303-2<??> is defined only in terms of the entire AP. Therefore, conformance requires that an implementation pass executable versions of all abstract test cases in clause 6.

E.13 Annex B Information object registration

The text in this subclause shall be used for the required normative annex B, with the appropriate part number inserted and without the indentation shown here.

To provide for unambiguous identification of an information object in an open system, the object identifier

{ iso standard 10303 part(3<??>) version(1) }

is assigned to this part of ISO 10303. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

E.14 Annex C Postprocessor input specification file names

The text in this subclause shall be used for the required normative annex C, with the appropriate part number inserted and without the indentation shown here.

The postprocessor input specifications for each test case is supplied electronically on magnetic media (floppy diskette). Table C.1 lists the file name of the postprocessor input specification that is associated with the postprocessor subclause(s) of a test case.

[Note: Fill the following table - one row for each postprocessor input specification.]

Table C.1 Postprocessor input specification file names

Subclause	Test case	File name
6.1.2	<Test_case_name>	<filename>

Subclause	Test case	File name
…	…	…
6.x.x	<Test_case_name>	<filename>

E.15 Annex D Excluded test purposes

Annex D is a required informative annex that shall contain the list of all test purposes derived from the AIM EXPRESS that have been deliberately excluded from the abstract test suite. The reasons that the test purposes were excluded shall also be provided. The test purposes shall be grouped into subclauses according to the reason for exclusion. If there are no excluded test purposes then the following text should appear in this annex, without the indentation shown here:

There are no AIM test purposes which have been excluded.

E.16 Annex E Bibliography

Use this annex to list referenced documents, including those parts of ISO 10303 that have not yet advanced in the ISO standardization process to be referenced normatively. The format is specified below.

1. ISO 10303-xx *Industrial automation systems and integration — Product data representation and exchange — Part xx: Xxxxx xxx xxx.*

E.17 Index

The index shall contain at least a listing of all the application objects and where they are found in the test purposes.

Annex F
(normative)

Abstract test case template

Each abstract test case shall be documented as a separate subclause of the abstract test suite part. Annex F.1 provides a template for each of the abstract test cases. The template is correct except that in the subclause numbering, the abstract test case clause number (clause 6) needs to replace the annex F clause number shown here. Text that needs to be replaced with actual text appropriate to the particular abstract test case or comments that should not be included are marked by angle brackets <this is an example>. Subclauses F.2 and F.3 provide a more detailed description of the template but are not part of the template.

F.1 [Abstract test case title]

Test case summary: <The test case summary is composed of two to three sentences about the general purpose of the particular abstract test case.>

Execution sequence: <This is an optional block that specifies details about the sequence of operations required for performing this test.>

Extra details: <This is an optional block that specifies information that is necessary for the abstract test case but for which there is no specified place for documentation.>

F.1.1 Preprocessor

Test purposes covered:

The following general test purposes are covered: g1, g2, and g3 <add other general test purposes as appropriate>. In the preprocessor input specification table of a test case, the numbers in column 1 (ignoring the part beyond the decimal point, if any), whose rows are not empty in column 2 (V), identify the AE test purposes covered by this test case.

[If other test purposes are covered by this preprocessor test case add the following text:]

The following other test purposes are covered: <list the other test purpose identifiers here separated by commas>.

Input specification: See Table 1

[All the entries in the table below the header row are example data for explanatory purposes only. The Table shall follow the formatting rules for tables defined in the SC4 Supplementary Directives.]

Table 1 Application Elements for [insert abstract test case title here]

Id	V	Application Elements	Value	Req

@1	*	Application_object1	#20	M
@2		Application_object1.attribute1	#20, “Value of attribute1”	S
@3	1	Application_object1.attribute2	#30	S
@4	*	Application_object1 to Application_object2 (role)	@100.1	M
@6		Application_object3 to Application_object1 (role)	@200	M
@100.1	*	Application_object2	#190	M
@101	*	Application_object2.attribute1	#200, <Figure 1>	C1
@5	*	Application_object1 to Application_object2 (role)	@1	M

[Insert supporting input specification data such as figure here.]

Constraints on Vaues:

[Insert a list of constraints referenced in the input specification table each one starting a new line and beginning with the label “C<number>: “ where <number> is an integer. Note: this section is only included if there are any constraints specified in the Req column of this table]

C1: [describe here any special constraints on changing the value for Application_object2.attribute1 (for the example data in Table 1 above as well as any effects those changes might have on other parts of the input specification and verdict criteria)]

Verdict criteria:

[Insert the list of preprocessor specific verdict criteria, each starting on a new line following this heading]

VC1: [specify the semantic verdict criterion for Application_object1.attribute2 (from the example data in Table 1 above)]

Execution sequence: [This is an optional block that specifies details about the sequence of operations required for performing this preprocessor test.]

Extra details: [This is an optional block that allows for entering information that is necessary to the preprocessor abstract test case but for which there is no specified place for documentation]

F.1.2 Postprocessor

[Note that more than one postprocessor input specification may be required to properly cover the possible range of AIM constructs that could be used to represent the application elements for a particular abstract test case. Hence there may be more than one postprocessor subclause in a single test case. If there are two or more postprocessor input specifications for an abstract test case, they shall be listed sequentially in separate subclauses, each titled "Postprocessor n" where n is a sequential integer which uniquely identifies the postprocessor input specification within the abstract test case. Each postprocessor subclause is divided into separate sections for the test purposes covered, the postprocessor input

specification reference, and the verdict criteria. When there is but one postprocessor input specification in the abstract test case, the sequential integer shall be dropped.]

Test purpose coverage:

The following general test purposes are covered: g1, g4, and g5 <add other general test purposes as appropriate>. The numbers in column 1 (ignoring the part beyond the decimal point, if any) of table 1, whose rows are not empty in column 2 (V), identify the AE test purposes covered in this test case. The following AIM test purposes are covered: <list the AIM test purpose identifiers separated by commas>.

[If other test purposes are covered by this postprocessor test case add the following text:]

The following other test purposes are covered: <list the other test purpose identifiers here separated by commas>.

Input specification 1:

[insert the first ISO 10303-21 or ISO 10303-12 (EXPRESS-I) representation of the input here]

Input specification 2:

[insert the second ISO 10303-21 or ISO 10303-12 (EXPRESS-I) representation of the input here (and so on as needed)]

Verdict criteria:

[Insert the list of postprocessor verdict criteria, each starting on a new line following this heading. The verdict criteria will be the same for all versions of the postprocessor input specification.]

VC1:

Execution sequence: [This is an optional block that allows for specifying details about the sequence of operations required for performing this preprocessor test.]

Extra details: [This is an optional block that allows for entering information that is necessary to the preprocessor abstract test case but for which there is no specified place for documentation]

F.2 Preprocessor input specification documentation syntax

The following productions define the lexical elements and grammar rules for each of the elements used in the preprocessor input specification table. The notation builds on the lexical elements and grammar rules defined in annex A. The primary rule for the Id column is 65; the primary rule for the V column is 70; the primary rule for the Application Elements column is 56; the primary rule for the Value column is 60; and the primary rule for the Req column is 68.

50. ae-aggregated-value = ‘(’ ae-value-form ‘)’ { ‘,’ [\n] ‘(’ ae-value-forms ‘)’ } .

51. ae-enumerated-value = simple-id .

52. ae-explicit-value = ae-string-value | logical-literal | real-literal | ae-enumerated-value | special-value | ae-named-value .

53. ae-explicit-value-set = '(' ae-explicit-value { ',' ae-explicit-value } ')' .

54. ae-named-value = simple-id '=' ae-explicit-value

55. ae-string-value = string-literal .

56. ae-spec = object-name | ao-attribute | appl-assertion | ao-categorisation .

57. ae-value-form1 = post-reference | ae-explicit-value | ae-explicit-value-set .

58. ae-value-form2 = post-reference ', ' (ae-explicit-value | ae-explicit-value-set) .

59. ae-value-form = ae-value-form1 | ae-value-form2 .

60. ae-value-spec = [ae-value-form | ae-aggregated-value | ao-reference] .

61. ao-attribute = object-name '.' attribute-name .

62. ao-categorisation = object-name '(as ' object-name ')' .

63. ao-reference = id-spec [{ ',' id-spec }] .

64. appl-assertion = object-name ' to ' object-name '(' role-description ')' .

65. id-spec = '@' test-purpose-id ['.' digits] .

66. post-reference = '#' digits [{ ', ' '#' digits }] .

67. range = digits '-' digits .

68. req-spec = ['M' | 'S' | 'C' digits] .

69. special-value = '<not_present>' | '<see' ('Figure' | 'Table') real-literal '>' .

70. verdict-spec = ['*' | (digits | range) { ',' (digits | range) }] .

F.3 Preprocessor input specification development

The preprocessor input specification table has specific rules on how to represent the input data requirements for the abstract test case. The table has a heading row in 11 point font shown in the template. If the table spans more than one page then the heading is repeated at the top of the table on each new page. The table caption appears above the table. If the table spans more than one page then the caption is repeated at the top of the page with the string "(continued)" appended. The caption for the last page of the table is appended with "(concluded)". The table is outlined with double lines. The rows do not have borders between them except for rows with application objects which have a double line border above them. Between the double lines, entries are placed in the following order:

— The application object. If the application object is instantiated as one of its subtypes, then the supertype application object component of this instance is labelled as a categorisation (see F.3.3).

— The attributes for the application object (if any)

— The assertions for the application object (if any)

— The subtype application object (if any)

— The attributes for the subtype application object (if any)

— The assertions for the subtype application object (if any)

— Continue with any other subtype application objects as necessary.

The entries in the table are in 10 point font. Further rules for the table are provided below.

F.3.1 Id column

The input specification Id column serves a dual purpose: an identifier for the row and a reference to the test purpose number. The identifier is constructed with an '@' symbol followed by the test purpose Id (associated with the AE in this row of the table). If there is more than one instance of an application object in the table, the identifier for that application object shall be followed by a "." and an integer which makes the input specification Id unique across all application objects in the table[4].

EXAMPLE 54 - An application object Application_object1 has two attributes, attribute1 and attribute2. The test purposes are defined as:

ae10 Application_object1 (see 6.1)
ae11 Application_object1 with attribute1 (see 6.1)
ae12 Application_object1 with attribute2 (see 6.1)

The input specification Id's would be @10, @11, and @12 respectively:

Id	V	Application Elements	Value	Req
@10	*	Application_object1	#20	M
@11	*	Application_object1.attribute1	#20, "Value of attribute1"	S
@12	*	Application_object1.attribute2	#30	S

EXAMPLE 55 - If there are two instances of Application_object1 (as defined in Example 54) in the table, the first instance would have an Id of @10.1 and the second instance would have an Id of @10.2.

[4] There is a proposal to allow an alphanumeric identifier in place of the integer after the ".". This proposal is still being reviewed by Qualifications for acceptance.

F.3.2 V column

The Verdict column identifies whether the associated application element is verdicted in this test case. A '*' indicates that the application element is verdicted using a derived verdict criterion according to the rules defined in annex D. An integer also indicates the application element is verdicted and is used to reference one of the specific verdict criterion (or range of verdict criteria) identified in the specific verdict criteria section of the preprocessor subclause. A blank in this column indicates the application element is not verdicted in this test case.

EXAMPLE 56 - Application_object2 is a complex object for which the general and derived verdict criteria do not suffice. A reference is placed in the V column to the two verdict criteria associated with that object:

Id	V	Application Elements	Value	Req
@20	1-2	Application_object2	#3020, <see Figure 1>	S

Specific verdict criteria:

VC1: The model realized by the IUT corresponds in shape to the model represented in Figure 1.
VC2: The radius of the central hole of the ring is 5.0 mm.

F.3.3 Application elements column

The Application Elements column lists the application elements and categorisations for this input specification. The following forms are used:

— Application objects: these are represented using the application object name (see @1 in Table 1). All application objects have a double line border above the row.

— Categorisations: these are represented as they are for the categorisation test purpose (e.g. Application_object1 as Application_object2). The categorisation is used as the representation of the supertype component of a complex instance of a supertype-subtype relationship. If there is more than one subtype categorisation instance present in the input specification, then there is a separate row for each categorisation test purposes covered.

— Application attributes: these are represented as the application object name followed by a "." followed by the attribute name (see @2 in Table 1). They are listed first under the application object. If an aggregate attribute has more than one element present there is only one row for the attribute and the value column shows the values for each of the elements enclosed in parentheses and separated with a ",". For clarity each element's value for the aggregate can appear on a separate line in the value column. No border appears above the row.

— Assertions: these are represented as they are in clause 4 of the application protocol followed by the role-description in parenthesis (see @4 in Table 1). They are listed after the attributes for the application object. Both the primary relationships and the inverse relationships are listed under the application object (see @4 and @6 in Table 1). The format of the relationship is : Application_object1 to Application_object2 (role) where 'role' is the role description from clause 4 of the AP (as used in the test purpose). For inverse relationships, the role string is the inverse role description. If a one to many assertion has more than one relationship present then there is only one

row for the assertion and the Value column is used to list each of the application object Id's that are related separated by a ",". No border appears above the row.

EXAMPLE 57 - An application object Application_object1 has a defined categorisation: Each Application_object1 may be an Application_object2 or an Application_object3. The object also has a defined attribute and has a related application assertion. Consider the following set of example resultant test purposes:

ae10 Application_object1
ae11 Application_object1 as Application_object2
ae12 Application_object1 as Application_object3
ae13 Application_object1 with attribute1
ae14 Application_object1 is forward related to zero Application_object4
ae15 Application_object1 is forward related to one Application_object4
ae16 Application_object1 is inverse related from zero Application_object4
ae17 Application_object1 is inverse related from one Application_object4
ae18 Application_object1 is inverse related from many Application_object4
...
ae20 Application_object2
ae21 Application_object2 with attribute2
...
ae30 Application_object3
...
ae40 Application_object4
ae41 Application_object4 with attribute3

An example simple instance representing an instance of the supertype Application_object1 as itself (which is not related to any other instance of Application_object4) appears as:

Id	V	Application Elements	Value	Req
@10	*	Application_object1	#20	M
@13	*	Application_object1.attribute1	#20, "Value of attribute1"	S
@14	*	Application_object1 to Application_object4 (is forward related to)	<not_present>	M

An example complex instance representing an instance of the subtype Application_object2 (which is also related to an instance of Application_object4) appears as follows:

Id	V	Application Elements	Value	Req
@11	*	Application_object1 (as Application_object2)	#20	M
@13	*	Application_object1.attribute1	#20, 'Value of attribute1'	S
@15	*	Application_object1 to Application_object4 (is forward related to)	@40	M
@20		Application_object2	#2020	M
@21		Application_object2.attribute2	#2020, 99.9	S
@40		Application_object4	#400	M

@41		Application_object4.attribute3	#400, ‘value of attribute3’	S
@17	*	Application_object1 to Application_object4 (is inverse related from)	@11	M

F.3.4 Value column

The value for the application element is listed in the Value column. There are several forms possible:

— The value column for an application object contains a link to the mirror AIM entity in the postprocessor input specification by referencing the entity instance name (#number).

— The value column for categorisations follows the rule for simple application objects above.

— The value column for an attribute contains a link to the mirror AIM entity in the postprocessor input specification by referencing the entity instance name. The link serves as a pointer for a test case realiser to identify where the mirror value is located that may be changed.

EXAMPLE 58 - A preprocessor input specification for ATS303 includes a Part application object with two of its attributes defined as follows:

Id	V	Application Elements	Value	Req
@115	*	Part	#1	M
@118	*	Part.part_nomenclature	#1, “Flat Ring Gasket”	S
@119	*	Part.part_number	#1, “11113”	S

An excerpt from the postprocessor input specification for this test case is as follows:

…
#1=PRODUCT(‘11113’,”Flat Ring Gasket’,’Description for part 11113’,(#2));
#2=MECHANICAL_CONTEXT(‘detailed design’,#3,’mechanical’);
…

The mapping table for AP203 maps Part.part_nomenclature into product.name and part_number into product.id. The Value column for Part.part_nomenclature and Part.part_number identifies the PRODUCT entity with entity instance name, “#1”, in the postprocessor specification as the AIM entity instance that defines the mirror values. Since they are suggested values, the #1 shall appear in the Value column.

— The value column for attributes which have simple values shall also include a representation of the value for the attribute. This value follows the entity instance name (see @118 and @119 in EXAMPLE 58). The value for an attribute is simple if it maps to a base type. If an attribute maps to an AIM entity type then the ATS developer must determine if there is a non-ambiguous way to represent the value using one of the following forms. Strings shall be enclosed in single quote marks. Numerical, boolean, and enumerated values shall not be enclosed in single quotes. Sets of more than one value shall be enclosed in parentheses and separated by commas (see rule 53 in F.2). For purposes of clarity the AIM entity attribute name followed by an “=” can be used before the

value (see rule 54 in F.2). If the attribute is not present in the input model then the keyword, "<not_present> shall be used. If the value is not simple then the mirror entity instance name from the first postprocessor input specification is required. Additionally an external reference to a figure or other explanatory text can also be used in the value column (see rule 68 in F.2).

— The values for aggregate attributes are enclosed in parentheses and separated by a "," (see rule 50 in F.2).

EXAMPLE 59 - The following are legal values for the Value column of an attribute:

#2020, TRUE
approved
‘’
‘Flat ring gasket’
1.000
#3120, (2.0, 2.5, 1.8)
(#100,2.0), (#200,3.0)
#100, #101, ‘John Smith’
first_name = ‘John’
(first_name = ‘John’, last_name = ‘Smith’)
<not_present>
#3030, <see Figure 1>

— The value column for assertions contains the Id number (@number) of the related application object. If the assertion is a one to many assertion with more than one application object relationship present, then the value column contains a list of the Id numbers separated by a ",". If the assertion is not present in the input model then the keyword "<not_present>" is used.

— Any preprocessor input specification that has more than one associated postprocessor input specification must provide an appropriate value for each postprocessor mirror. Each value is preceded by the identification number of the appropriate postprocessor specification followed by a ': '. The values are separated by blank space or a newline.

EXAMPLE 62 - The following are legal entries for the Value column of an attribute row in a preprocessor input specification that is associated with two postprocessor input specification mirrors:

1: #200,'Value1' 2: #300, 'Value1' , or

1: #200, 'Value1'

2: #300, 'Value1'

F.3.5 Req column

The Req column specifies the requirements on the values in the Value column. Values in the value column shall be one of mandatory, suggested, or constrained (denoted by a ‘M’, ‘S’, or ‘Cn’ in the Req column). Values in an abstract test case are normally suggested values which means the test case realiser is free to change those values provided that the new value is still of the same type. Values shall be mandatory due to requirements of the standard (EXPRESS rules, mapping table requirements, or test purposes). Values shall be constrained if they are not mandatory but any changes to the value may cause non-obvious changes in other parts of the test case. Changes to the values may also be constrained by

the test case to a defined set of allowable alterations. *All values in the preprocessor input specification shall be suggested unless one of the following rules applies (rules for mandatory values take precedence over rules for constrained values).*

a) The value for an assertion present in the input specification table is mandatory.

Rationale: Assertions describe the structure of the file. To change their value implies changing the structure of the file. Although it is conceivable that one could trace through an assertion and all its related elements and determine whether it could be changed without affecting any other verdict in the file, such a task would be extremely laborious. It would also make maintenance extremely difficult as every re-assignment of verdicts or changes to the test case would force a re-assessment of each assertion. Hence it is strongly recommended that all assertions be labeled as M. Note that not all assertions for an application object appear in the input specification table. This rule only applies to those assertions actually listed in the table.

b) The value for an application object which has attributes is mandatory.

Rationale: This is merely a convention. The value requirement for an application object with attributes is defined by the value requirements for each of the attributes.

c) The value for an application object which has no attributes follows the same rules as attributes below.

Rationale: The Value column for an application object without any attributes defines the value for the object. Therefore the Req column shall specify the constraints (if any) on changes to the value by the test case realiser.

d) The value for an attribute which is verdicted and which has a test purpose of the form, "Application_object with attribute = value" is mandatory.

Rationale: Attributes which are verdicted and which have a restricted set of values or are mapped to AIM attributes of type, ENUMERATION, BOOLEAN, or LOGICAL will have test purposes identified for each of the allowed values. In order for any of those test purposes to be correctly verdicted in a test case the value assigned cannot be changed, and therefore it is mandatory.

e) The value for an attribute which is verdicted and which has a value restricted by a value constraint in the mapping table is mandatory.

Rationale: Attributes which are verdicted and which have values restricted by a value constraint in the mapping table shall be verdicted with those values.

f) The value for an attribute which is verdicted but whose value is <not_present> is mandatory.

Rationale: One of the test purposes for an optional attribute uses the form, “Application_object with attribute not present”. In order to verdict this test purpose it is mandatory that the attribute not be present hence there can be no other allowed "value" for the attribute.

g) The value for an attribute is constrained when the attribute is verdicted and when changes to the value might cause changes to other parts of the test case which may not be obvious.

Rationale: Marking a value as constrained serves as a warning to the test case realiser that changes to this value may cause changes to other parts of the test case (such as other parts of the input model or other verdict criteria) which may not be obvious. The constraint referenced by this column shall clearly explain what parts of the test case might be impacted by a change in the value.

h) When the semantic value defined in the table for an attribute has specific domain importance the value is constrained.

Rationale: ATS developers may determine that a certain value should be assigned to an application element since the value tests for conditions which are important to the domain of the AP but which are not necessarily explicitly identified by the standard. For example, in AP203 the ATS developers wanted a range of different kinds of geometry to be used in the test suite to cover geometry of interest to the domain but not called out in the information requirements. If these values were left as suggested values, a test case realiser would be free to substitute a completely different set of geometry which would no longer serve the purpose of the domain expert's test suite.

The Req column specifies whether the value is mandatory (‘M”) , suggested (‘S’) or constrained (‘Cn’ where n is replaced by an integer) according to the rules defined above. All the Cn’s in the table shall have a unique integer since they serve as references to one or more notes detailed in the Constraints on Values section of the test case. The explanatory text associated with the constraint shall clearly identify the reason for the constraint and provide sufficient description of what aspects of the test case would have to be changed if the value were changed. If the value can only be changed within certain bounds or according to certain criteria then those shall be clearly identified. For example a solid ring might be used as the test shape in ATS303. Certain dimensions of the ring might be changed but only so the ring still remains circular with a hole in the center.

F.3.6 Pruning the table

In order to reduce the size of the input table, entries shall be removed. Those rows where the application element is not verdicted (the V column is blank) and whose value in the Value column is <not_present> shall be removed from the table.

Annex G
(informative)

Bibliography

*ISO 10303-12, Industrial automation systems and integration - Product data representation and exchange - Part 12: Descriptive methods: The EXPRESS-I language reference manual*ISO 10303-201: 1994 Application Protocol: Explicit Draughting

ISO 10303-203 :1994 Application Protocol: Configuration controlled 3d designs of mechanical parts and assemblies

ISO TC184/SC4 "Supplementary directives for the drafting and presentation of ISO 10303," version 2.3, (Editing N48), 15 March 1995.

ISO TC184/SC4 "Guidelines for the Development and Approval of ISO 10303 Application Protocols," version 1.1 (WG4 N103), 30 November 1993.

Industrial Technology Institute, "STEP Test Notations: Requirements and Capabilities Survey", technical document submitted to ISO TC184/SC4/WG6, April 1992.

Industrial Technology Institute, "Commands for a STEP Abstract Test Notation'', technical document submitted to ISO TC184/SC4/WG6, July 1992.

Annex H
(informative)

Approval of abstract test suites

This annex presents a process for reviewing early versions of abstract test suites and a checklist for the approval of abstract test suites. The purpose of the reviews is to provide guidance to abstract test suite developers as they develop abstract test suites. The purpose of the checklist is to provide guidance to abstract test suite developers and reviewers on what a completed abstract test suite shall contain.

H.1 Abstract test suite document contents requirements

The following are the requirements on the contents of an abstract test suite for release as a committee draft, draft international standard, or final draft international standard:

a) No defects[5)] with respect to the "Guidelines for abstract test suite development" or ISO 10303-33.

b) Guidelines annex E text requirements are met.

c) Checklists contained below satisfactorily completed.

d) No defects with respect to Supplementary Directives (the most recent version released by the Editing committee).

e) No defects with respect to any other ISO 10303 part at committee draft level or higher.

f) A complete set of test purposes is defined.

H.1.1 Abstract test suite approval checklist

The following subclauses form a checklist for approving an abstract test suite part.

H.1.1.1 ISO document requirements

In general, the abstract test suite is required to meet the documentation requirements of the current version of the SC4 Supplementary directives. More specifically:

a) Is the table of contents present and correct?..... YES/NO

b) Is the foreword present and correct?..... YES/NO

---

5) See SEDS process (N308) for definition of defect.

c) Is the introduction present and correct?..... YES/NO

d) Are the formatting details of the clauses correct? ..... YES/NO

e) Are the normative references present and correct? ..... YES/NO

f) Is the list of required definitions present and correct? ..... YES/NO

g) Is the index present and correct? ..... YES/NO

For approval to be granted, items a, b, c, d, e, f, and g above must have 'YES' answers.

H.1.1.2 List of test purposes

Test purposes are required in the abstract test suite. See clause 7 for more specific information on the sources of test purposes and annexes A, B, and C for information on the development and documentation of test purposes.

a) Has the list of test purposes been included (see 6.5)? ..... YES/NO

b) Does each application object in the AP have a corresponding test purpose in the ATS and the appropriate additional test purposes for any categorisations (B.1)? ..... YES/NO

c) Does each application object in the AP have test purposes corresponding to its attributes (B.2)? YES/NO

d) Is the order of the application object test purposes the same as the order of in which the application objects appear in the information requirements of clause 4.2 (E.9)?..... YES/NO

e) Does each assertion in the AP have test purposes corresponding to the relationship it defines (B.3)?..... YES/NO

f) Does each AIM entity have a corresponding test purpose (C.1)? ..... YES/NO

g) Does each AIM entity have test purposes corresponding to its attributes (C.2)?..... YES/NO

h) Do all test purposes derived from the AEs and AIM of the AP meet the syntax requirements for test purposes (annex A)? ..... YES/NO

i) Does each test purpose appear in at least one abstract test case along with an associated verdict criterion?..... YES/NO

j) Are all test purposes not being addressed in an abstract test case with an associated verdict criterion documented in an informative annex (E.15)?..... YES/NO

For approval to be granted, items a, b, c, d, e, f, g, h, i, and j above must have 'YES' answers.

H.1.1.3 General test purposes and verdict criteria

General test purposes and verdict criteria are recommended where appropriate. See subclause 6.6 for more specific information on general test purposes and verdict criteria.

a) Are general test purposes present (6.6)? ..... YES/NO

1) If YES, is each general test purpose present unambiguous? ..... YES/NO

2) If YES, does each general test purpose present relate to one or more conformance requirements?..... YES/NO

3) If YES, is each general test purpose explicitly related to some part of clause 6 of the associated AP? ..... YES/NO

4) If YES, is a general verdict criterion associated with each general test purpose?..... ..... YES/NO

5) If YES, is each general verdict criterion present unambiguous? ..... YES/NO

6) If YES, does each general verdict criterion reference the test purpose identifier(s) of the related test purpose(s) ? ..... YES/NO

7) If YES, are the abstract test cases to which each general test purpose applies clearly identified? ..... YES/NO

For approval to be granted, if item a above has a 'YES' answer, sub-items 1, 2, 3, 4, 5, 6 and 7 must also have 'YES' answers.

H.1.1.4 Abstract test cases

a) Has a checklist for each abstract test case (H.1.2) been completed satisfactorily? ..... YES/NO

b) Does the AP contains basic tests (4.4)? ..... YES/NO

1) If YES, are the basic tests listed first? ..... YES/NO

For approval to be granted, item a above must have a 'YES' answer and, if item b has a 'YES' answer, sub-item 1 must also have a 'YES' answer.

H.1.1.5 Annexes

a) Has normative annex A, Conformance classes, been included (E.12)? ..... YES/NO

b) Is annex A complete (E.12)? ..... YES/NO

c) Has normative annex B, Information object registration, been included (E.13)? ..... YES/NO

d) Is annex B complete (E.13)? ..... YES/NO

e) Has normative annex C, Postprocessor input specification file names been included (E.14)? YES/NO

f) Is annex C complete (E.14)? ..... YES/NO

g) Has informative annex D, Excluded test purposes, been included (E.15)? ..... YES/NO

h) Is annex D complete (E.16)? ..... YES/NO

For approval to be granted, items a, b, c, d, e, f, g, and h above must have 'YES' answers.

H.1.2 Abstract test case approval checklist

The following subsections form a checklist for approving the individual abstract test cases within an abstract test suite. For approval for Committee Draft status, all abstract test cases must be present and correct as outlined by this checklist and supported by clause 6.7 and annex F.

H.1.2.1 Test case identifier

a) Is the test case a separate subclause of the abstract test suite document (6.7)? ..... YES/NO

b) Is the subclause identifier present as the subclause title (6.7.1)? ..... YES/NO

c) Is the subclause identifier unique (6.7.1)? ..... YES/NO

For approval to be granted, items a, b and c above must have 'YES' answers.

H.1.2.2 Test case summary

a) Is a test case summary present (6.7.2)? ..... YES/NO

For approval to be granted, item a above must have a 'YES'.

H.1.2.3 Input specifications

a) Is there at least one test purpose covered by the abstract test case (4.2)? ..... YES/NO

b) Is the preprocessor input specification present (6.7.3, F.3)? ..... YES/NO

c) Does the preprocessor input specification provide enough information to unambiguously lead to an executable test case (hardcopy for preprocessors, ISO 10303-21 exchange structure or Express-I model instantiation for postprocessors) (6.7.3, F.2, and F.3)? ..... YES/NO

d) Is the preprocessor input specification correctly documented in a table plus any needed ancillary information?(F.3) ..... YES/NO

e) Do the postprocessor input specifications provide enough information to unambiguously lead to an executable test case which meets the test purposes covered by this abstract test case (6.7.4)? ..... YES /NO

f) Are the postprocessor input specifications correctly referenced and included in a magnetic media (6.7.4.2)? ..... YES/NO

g) Are the postprocessor input specifications correctly documented in ISO 10303-21 exchange structures or Express-I model instantiations (6.7.4)?..... YES/NO

For approval to be granted, items a, b, c, d, e, and f above must have 'YES' answers.

H.1.2.4 Specific verdict criteria

a) Are specific verdict criteria present (4.3 6.7.3.4, and 6.7.4.3)? ..... YES/NO

1) If yes, is each specific preprocessor verdict criterion unambiguous (6.7.3.4)? ..... YES/NO

2) If yes, does each specific preprocessor verdict criterion evaluate to PASS, FAIL, or INCONCLUSIVE (7.6.3.4)? ..... YES/NO

3) If yes, is each specific postprocessor verdict criterion unambiguous (6.7.4.3)? ..... YES/NO

4) If yes, does each specificpostprocessor verdict criterion evaluate to PASS, FAIL, or INCONCLUSIVE (6.7.4.3)? ..... YES/NO

For approval to be granted, if item a above has a 'YES' answer, sub-items 1, 2, 3, and 4 must have 'YES' answers.

H.1.2.5 Additional test case information

Additional test case information includes a collection of optional information blocks that are addressed in 8.2.6 of the ATS guidelines

a) Are extra details on the test present (6.7.4.4)? ..... YES/NO:

1) If YES, are the details clear and unambiguous?..... YES/NO

b) Is a sequence of construction for the test present (6.7.4.4)? ..... YES/NO

1) If YES, is the sequence clear and unambiguous? ..... YES/NO

For approval to be granted:

— If a has a 'YES' answer, its sub-question 1 must also have a 'YES' answer;

— If b has a YES' answer, its sub-question 1 must also have a YES' answer.

Index